UNIVERSIDAD AUTONOMA DE MADRID

Escuela Politécnica Superior

20|21

Trabajo fin de grado

ENCODE: módulo de seguimiento de estudiantes

Eduardo Rico Mercader

Escuela Politécnica Superior
Universidad Autónoma de Madrid
C\Francisco Tomás y Valiente nº 11

excelencia Campus Internacional UAM CSIC+

www.uam.es

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

ENCODE: módulo de seguimiento de estudiantes

Autor: Eduardo Rico Mercader
Tutor: Alejandro Sierra Urrecho

noviembre 2020

Eduardo Rico Mercader
ENCODE: módulo de seguimiento de estudiantes

Eduardo Rico Mercader
C\ Francisco Tomás y Valiente Nº 11

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

A mis padres y a mis hermanos

La vida es como montar en bicicleta.
Para mantener el equilibro hay que seguir pedaleando.

Albert Einstein

Agradecimientos

Para empezar, quisiera expresar mis agradecimientos a mi tutor de este trabajo, Alejandro Sierra Urrecho, por la propuesta del trabajo así como su atenta supervisión. También quiero agradecer a que me animara a comenzar el trabajo en el verano de 2020, para que así estuviera listo antes del comienzo de las clases y se le pudiera sacar el máximo partido.

Quiero dar las gracias a Abel Frías, que desarrolló la primera versión de ENCODE, y que además ha colaborado para la integración de los nuevos cambios realizados. También quiero reconocer a todas las personas que han colaborado anteriormente en ENCODE y que han permitido que este proyecto sea posible.

Por supuesto, agradezco la labor del personal docente que han facilitado mi formación y gracias a los cuales he adquirido los conocimientos necesarios para poder realizar un proyecto de esta complejidad. También quisiera mencionar al personal no docente de la universidad, por estar siempre ahí cuando se les necesita.

Finalmente, agradezco a mi familia todo el apoyo recibido a lo largo de la carrera, y por animarme a seguir adelante en los momentos más complicados.

Resumen

La aplicación ENCODE, desarrollada en 2018 como resultado de un Trabajo de Fin de Grado de la EPS, es una herramienta que permite la presentación de contenidos relacionados con programación. También permite la realización de preguntas para que los estudiantes se puedan poner a prueba a medida que aprenden, y dichas preguntas se repiten con cierta frecuencia para contribuir al aprendizaje a largo plazo. Sin embargo, la aplicación no dispone de una forma de consultar los resultados generados con su uso sin la ayuda de una herramienta externa.

Este TFG consiste en la realización de un módulo de estadísticas en ENCODE, para mostrar ciertos parámetros relacionados con las respuestas de los estudiantes a las preguntas que ofrece la aplicación. La información se muestra agrupada de diferentes maneras, y se presenta en forma de tablas y gráficos.

Este módulo permite a los profesores acceder al progreso de los estudiantes con datos actualizados de sus respuestas. También se incluyen varias opciones de filtrado de la información, para permitir mostrar datos de un subconjunto de usuarios o de tipo de preguntas.

Con la realización de este proyecto, un profesor puede hacer un seguimiento de los resultados de los estudiantes durante un curso. También permite a cualquier estudiante ver algunos de sus propios datos y compararlos con los datos medios de su clase.

Este documento comienza explicando el funcionamiento de la aplicación preexistente y la manera en la que gestiona la información. A partir de esto, se exponen las decisiones de diseño tomadas para la creación del nuevo módulo, y la forma en la que se ha implementado, incluyendo los cambios que ha habido que realizar.

Palabras clave

Aplicación web, estadísticas, análisis de datos, repaso espaciado, Django

Abstract

The ENCODE application, developed in 2018 as a result of a thesis at the EPS, is a tool that allows the presentation of educational content related to programming. It also makes questions that allow the users to test themselves as they learn, and said questions are repeated with certain frequency to allow long-term learning. However, the application does not include a way to analyse the results generated by its use without relying on an external tool.

This project consists on the implementation of a statistics module in the ENCODE project, to show certain parameters related to the student answers for the questions offered by the application. The information is grouped in different ways and displayed in the form of tables and charts.

This module allows the teachers to access the progress of the students with up-to-date information of their answers. There are some options for filtering information, to allow showing data from a subset of users or question types.

With the completion of this project, a teacher has the ability of overviewing the students' results during a course. It also allows any student to view some of their own data and compare them with the overall of their class.

This document is started by explaining the behaviour of the preexisting application and the way the information is managed. Afterward, the design choices required for the creation of the new module will be presented, as well as the way it has been implemented, including the required changes that had to be effected.

Keywords

Web application, statistics, data analysis, spaced repetition, Django

Índice

Listas

Lista de figuras

Lista de tablas

1

Introducción

ENCODE es una plataforma creada en 2018 como resultado de un TFG [1], con el objetivo de presentar contenidos relacionados con programación a los estudiantes de la EPS de la UAM.

Esta plataforma también permite la realización de preguntas para que los estudiantes puedan ponerse a prueba y favorecer su aprendizaje mediante la repetición de las mismas. Sin embargo, el profesor no dispone de una manera directa de poder analizar esta información.

Este Trabajo de Fin de Grado consiste en la creación de un módulo de estadísticas en ENCODE, para permitir el análisis de las preguntas respondidas y de otras interacciones de los usuarios, que permita realizar un mejor seguimiento de su avance.

1.1. Motivación

La principal motivación de este trabajo es permitir que los administradores de ENCODE puedan supervisar gráficamente el progreso de los estudiantes en la plataforma. Hasta ahora, la aplicación contaba con un registro de las preguntas respondidas, pero para su análisis, era necesario descargar los datos y utilizar una herramienta externa, lo cual requería mucho tiempo y trabajo manual.

Mediante la realización de un módulo incorporado en la propia aplicación de ENCODE, se permitirá poder analizar los datos que tiene la aplicación en cada momento, de modo que los datos estarán actualizados y se reducirá considerablemente el tiempo requerido. Además, se podrá filtrar la información para un subconjunto de estudiantes, cursos, y/o tipo de preguntas.

También se pretende que los estudiantes puedan ver una información general de su propio progreso y poder compararlo con el resto de la clase.

1.2. Organización de la memoria

Este documento dispone de los siguientes apartados: introducción, situación inicial, diseño, implementación, integración, pruebas, conclusiones y trabajo futuro.

Teniendo en cuenta que este proyecto consiste en la ampliación de otro ya existente, se comenzará explicando el estado de la aplicación antes del comienzo de este trabajo. Se describirá el funcionamiento de la aplicación y la forma en la que se podían analizar los datos en la versión original.

A continuación, se explicará cómo se ha decidido realizar esta ampliación del proyecto. Se describirán los requisitos de la funcionalidad a realizar, la información que se mostrará y una idea general de cómo se presentará esta información.

Después se continuará explicando los detalles de implementación. Se destacarán los cambios necesarios en la base de datos y los módulos ya existentes, y se mostrarán detalles de cómo ha quedado el resultado final.

Luego se describirá el proceso realizado de integración y las pruebas realizadas, junto con los cambios que se han realizado tras la integración inicial.

Finalmente, se comentarán unas conclusiones de la realización del proyecto y se expondrá el trabajo futuro de la aplicación.

2

Situación inicial

A continuación se explica el estado inicial del proyecto antes de la realización de este trabajo.

Por un lado, se expondrá el funcionamiento general de ENCODE y la manera en que gestiona la información, que será relevante para la posterior explicación de su uso en este proyecto. Por otro lado, se explicará la forma en la que se podían analizar los datos, razonando cuáles serían las ventajas de realizar un módulo nuevo, lo cual motiva la realización de este trabajo.

2.1. Trabajo previo

2.1.1. Descripción de la aplicación

ENCODE es una aplicación accesible mediante el uso de un navegador de internet a través del enlace `https://encode.eps.uam.es`. La aplicación ha sido desarrollada en *Django*, un framework de desarrollo web que permite la creación de este tipo de aplicaciones. Este entorno facilita la estructuración de la aplicación siguiendo el patrón de diseño modelo-vista-controlador, además de simplificar el desarrollo en varios aspectos.

Cualquier persona puede registrarse en la página y con ello acceder a los contenidos. Actualmente, dispone de tres cursos, *Programación en C*, *Programación en Kotlin* y *Programación en Android*. Estos contenidos se utilizan para que los estudiantes de *Programación 1* y *Desarrollo de aplicaciones para dispositivos móviles* puedan aprender los contenidos de dichas asignaturas. Dado que la aplicación está orientada a este fin didáctico, se hablará indistintamente de *Usuarios* y *Estudiantes*, así como de *Administradores* y *Profesores*.

Los administradores tienen acceso a un módulo *editor*, como se muestra en la figura 2.1, en el que gestionar los contenidos que verán los usuarios. Por otro lado, los estudiantes sólo pueden utilizar el módulo *tutorial*, en el que visualizar los contenidos. Obviamente, un administrador también puede acceder al *tutorial*.

Los cursos disponibles en el tutorial se dividen en unidades, y cada unidad, en una serie de objeti-

vos. Cada objetivo está pensado para presentar una píldora informativa, que puede incluir imágenes, código o incluso preguntas. Se muestra un ejemplo en la figura 2.2.

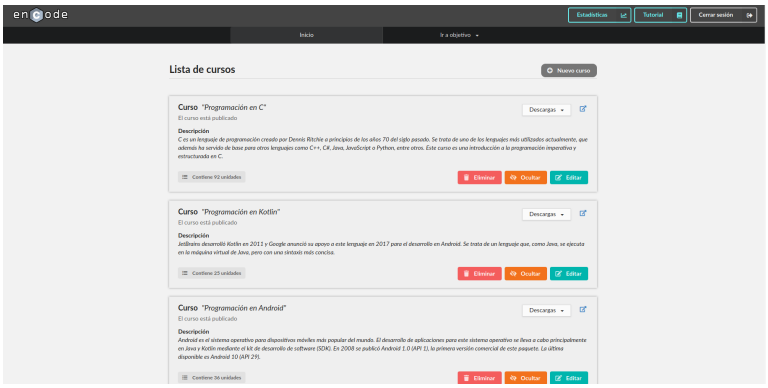


Figura 2.1: Página principal del editor de cursos

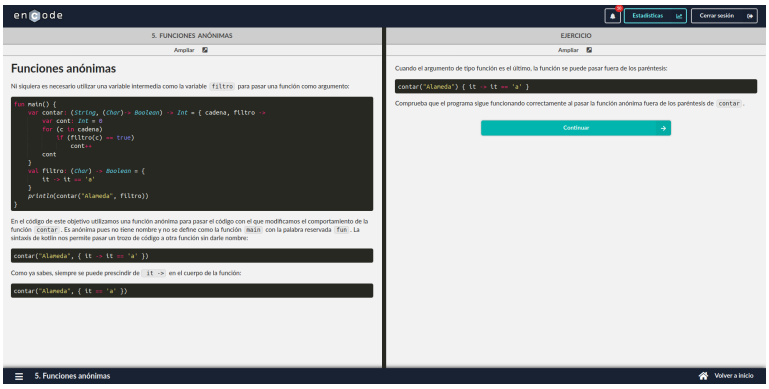


Figura 2.2: Ejemplo de objetivo del tutorial

Un usuario se puede inscribir en cualquiera de los cursos existentes, y avanzar a su propio ritmo. Las unidades y objetivos se desbloquean a medida que se avanza, pero también se permite repasar los contenidos ya vistos en cualquier momento.

Cuando un usuario termina todos los objetivos de una unidad, se le realizan una serie de preguntas en relación con lo que ha aprendido. Las preguntas se pueden utilizar para mostrar el entendimiento de un concepto o para razonar la respuesta a un problema. Se distinguen dos tipos:

- Preguntas abiertas

  Tratan de que el estudiante recuerde un concepto teórico o sepa dar la solución a un problema sencillo. Incluyen una solución que el usuario puede ver después de pensar su propia respuesta. Tras ver dicha solución, el usuario realiza una autoevaluación: “Difícil”, “Dudo” o “Fácil”, dependiendo de lo bien que supiera la respuesta. Se muestra un ejemplo en la figura 2.3.

- Preguntas tipo test

  Preguntas con varias opciones de respuesta. Tras seleccionar una opción, se puede ver cuál era la respuesta correcta, y ocasionalmente, puede ver una justificación de dicha opción. Se registra si la pregunta se acierta o falla. Se muestra un ejemplo en la figura 2.4.

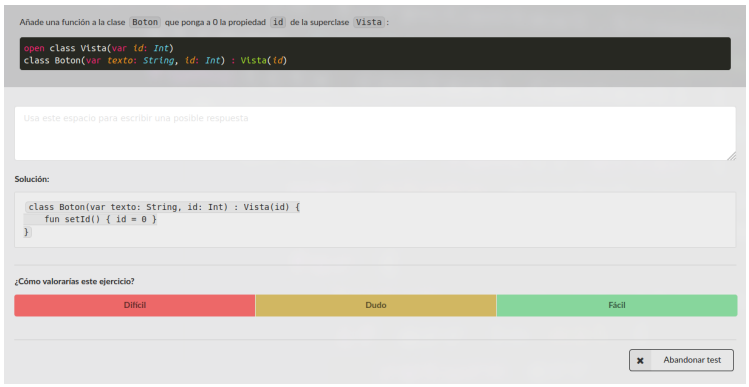


Figura 2.3: Ejemplo de pregunta abierta

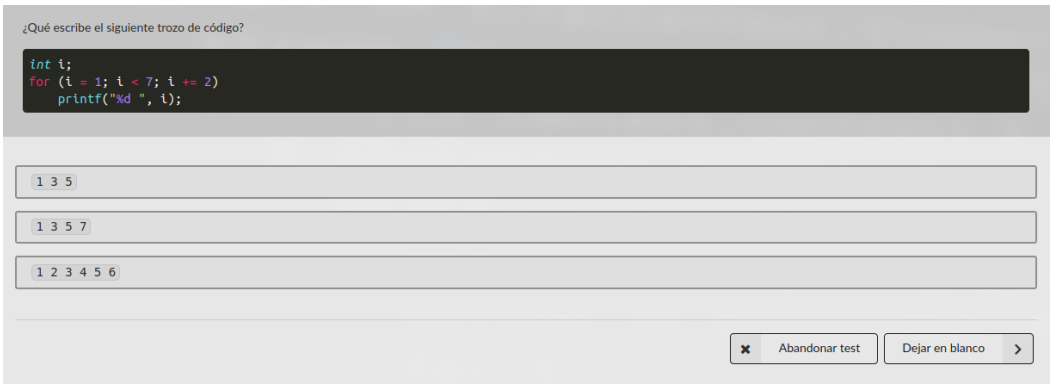


Figura 2.4: Ejemplo de pregunta tipo test

Mientras que las preguntas que forman parte de los objetivos no se muestran de ninguna otra forma, las preguntas de final de unidad se pueden repetir, y esta es una de las características principales de ENCODE. Su funcionamiento consiste en el aprendizaje por repaso espaciado. Esto consiste en la repetición de los conceptos aprendidos, en forma de preguntas, en el momento apropiado para evitar que se olvide.

El algoritmo utilizado es una adaptación de SM-2, un algoritmo que fue desarrollado para la aplicación SuperMemo [2]. Este algoritmo se utiliza en otras aplicaciones que hacen uso del repaso espaciado, como por ejemplo Anki [3].

Para cada pregunta que responda un usuario, se guarda una *tarjeta* con una serie de parámetros que dependen del resultado que haya tenido el usuario al responder o autoevaluarse. Los parámetros más importantes con *intervalo* y *facilidad*. El *intervalo* determina el número de días que pasa desde la realización de una pregunta hasta su repetición. Este valor se recalcula cada vez que se responde la pregunta. El parámetro *facilidad*, que depende de los resultados del usuario al responder, ajusta el cálculo del siguiente intervalo.

La primera vez que el usuario entra en la aplicación cada día, se le ofrece realizar las preguntas que tenga pendientes, o también puede acceder mediante un botón disponible en el menú principal.

2.1.2. Clases principales

El entorno Django permite crear una serie de clases que se conviertan directamente en tablas de una base de datos SQL, y la realización de consultas en alto nivel. En este sentido, los términos “clases”, “tablas” y “entidades” se refieren al mismo concepto.

Las clases del proyecto previo estaban estructuradas en los dos módulos principales, editor y tutorial, dependiendo de donde se hace más uso de cada clase. En cualquier caso, todas son accesibles desde cualquier módulo. Además, se utiliza la clase *User* que provee el framework Django, para la gestión de los usuarios. La figura 2.5 muestra un esquema general de las clases, y la tabla 2.1 muestra una descripción general del funcionamiento de las mismas.

2.2. Análisis de los datos

Hasta ahora, para analizar los datos, el administrador tenía que utilizar una herramienta externa. Se había realizado una opción, disponible en el módulo editor, que permitía descargar la información de todos los registros de respuestas en formato CSV. Sin embargo, la descarga requiere mucho tiempo (del orden de varios minutos) y tiene que realizarse de nuevo cada vez que se quieren actualizar los datos. Además, a medida que crezca la aplicación, se registrarán más preguntas y los tiempos se volverán cada vez mayores.

Clases	***Características***
Course, Unit, Objective, ObjectiveCode	Los cursos se componen de unidades, y las unidades de objetivos. Los objetivos pueden tener código de programación para mostrar.
StudentCourse	Relaciona estudiantes con cursos, cada objeto de esta clase representa un estudiante matriculado en un curso.
UnitTest, UnitTestAnswer	UnitTest sirve para las preguntas asociadas a las unidades, y que se repiten mediante el algoritmo de repaso espaciado. En el caso de las preguntas cerradas, UnitTestAnswer contiene los datos de las posibles respuestas.
FlashCard	Representa las tarjetas para el repaso de preguntas. Dispone de una serie de parámetros, como se ha explicado anteriormente, para gestionar el momento en que se debería repetir una pregunta.
FlashCardHistory	Consiste en un registro generado cada vez que un usuario responde una pregunta. Esto permite saber, además de la respuesta seleccionada, los valores de los parámetros de la correspondiente *FlashCard* en el momento del registro. Esta información es útil para analizar el progreso de un estudiante a lo largo del tiempo, ya que se puede ver la evolución de los parámetros de la *FlashCard* asociada. La aplicación guarda estos registros y permite su descarga, pero no los utiliza de forma directa.
Test, TestAnswer	*Test* representa las preguntas que pueden estar contenidas en los objetivos, que siempre son preguntas cerradas, y *TestAnswer* gestiona las posibles respuestas. Estas preguntas no se repiten fuera de los objetivos.
StudentEvent	Guarda eventos de los estudiantes en relación con su interacción con los objetivos, para registrar las visitas, respuestas a preguntas y subidas de archivos. Nota: esta clase fue añadida con posterioridad al inicio del proyecto por parte de otro desarrollador, pero al ser desarrollado de forma independiente, se considera parte del proyecto previo.

Tabla 2.1: Jerarquía de clases al comenzar el proyecto

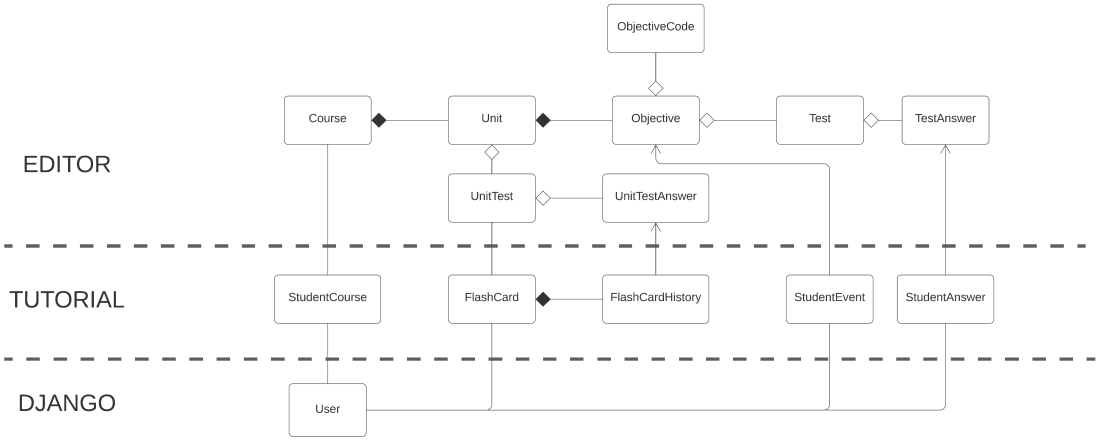


Figura 2.5: Estado inicial de la jerarquía de clases

Por ello, se propone la realización de un nuevo módulo en la aplicación, que permita poder analizar la información de las respuestas de los estudiantes a las preguntas sin depender de una herramienta externa. De esta manera, se podrá obtener información útil y actualizada en base a una serie de parámetros que se decidan de antemano. También será posible filtrar la información para un subconjunto de usuarios de la aplicación, así como los cursos y las etiquetas de las preguntas, para obtener información específica.

3

Diseño

Una vez entendido el proyecto de partida de la aplicación, y la necesidad que se pretende cubrir, se procede a explicar cómo se han decidido realizar las mejoras.

Para empezar, los cambios se realizarán mediante un nuevo módulo en la aplicación, como se ha comentado. Esto permite separar la lógica de negocio, los documentos que mostrar al usuario, y la definición de las clases que gestionan la información, del resto de la aplicación. Aunque se trate de simplemente una separación conceptual, esto dejará mejor organizado el código de la aplicación.

A continuación se explican las decisiones generales del diseño del módulo. Primero se listarán los requisitos que deberá cumplir el módulo. Posteriormente, se explicará qué clase de datos se intentarán obtener, y se detallará la forma en la se podrá filtrar la información.

3.1. Requisitos

Desde el comienzo se establecieron una serie de requisitos, que se fueron incrementando y detallando a medida que avanzaba el proyecto. A continuación se muestran los requisitos finales del módulo.

RF-1.– Mostrar información de los parámetros de las respuestas realizadas por los estudiantes en la aplicación.

- **RF-1.1.–** Mostrar la información en forma de tabla.
- **RF-1.2.–** Permitir agrupar esta información por cursos, etiquetas, estudiantes y preguntas. Para cada conjunto, mostrar los valores medios de los parámetros mostrados.
- **RF-1.3.–** Separar los datos de etiquetas, estudiantes y preguntas en distintos cursos. Por ejemplo, para ver el rendimiento de un estudiante en cursos distintos, o para distinguir etiquetas de varios cursos.

RF-2.– Mostrar gráficos para ilustrar la información.

- **RF-2.1.–** Mostrar un gráfico temporal con la distribución de la realización de respuestas a lo largo del tiempo.
- **RF-2.2.–** Mostrar una serie de histogramas con los parámetros numéricos de la información relacionada con las etiquetas y estudiantes.
- **RF-2.3.–** Mostrar un histograma representando los valores del parámetro *intervalo* de las preguntas

de los usuarios.

RF-2.4.– Mostrar un diagrama de barras con el parámetro de la facilidad media de las distintas etiquetas.

RF-3.– Permitir filtrar la información para un conjunto de cursos, usuarios, y etiquetas de preguntas.

RF-3.1.– Permitir guardar una lista de usuarios para poder facilitar el filtrado.

RF-4.– Mostrar a los estudiantes una información distinta que al administrador.

RF-4.1.– Mostrar información de su progreso en los cursos y el porcentaje de aciertos, en lugar de parámetros específicos del algoritmo que resulten difíciles de comprender.

RF-4.2.– Filtrar automáticamente para mostrar sólo los datos propios.

RF-4.3.– Mostrar datos medios de los estudiantes del curso que esté haciendo el estudiante.

RF-5.– Mostrar información de los eventos de objetivos.

RF-5.1.– Mostrar una tabla con el número de eventos producidos.

RF-5.2.– Mostrar un gráfico con la distribución de los eventos a lo largo del tiempo.

3.2. Análisis de las respuestas

Respecto a las preguntas, interesa analizar varios de los parámetros empleados en el algoritmo de repaso espaciado, principalmente:

- Valor de la calidad de las respuestas. Las preguntas abiertas se pueden autoevaluar como “Difícil”, “Dudo” o “Fácil”, lo cual se almacena con los valores 0, 3 y 5, respectivamente. En el caso de las preguntas cerradas, fallar es equivalente a “Difícil” y acertar es equivalente a “Fácil”.
- Valor de la facilidad de las tarjetas. Según el algoritmo utilizado, *facilidad* toma un valor inicial y se ajusta dependiendo de la *calidad* de las respuestas. Se pretende ver cómo evoluciona este valor tras las respuestas realizadas.
- Valor del intervalo de las tarjetas. También se busca conocer los intervalos de una pregunta hasta que se repite de nuevo. Cuanto mayor sea la *calidad* de una respuesta, mayor será el intervalo hasta la siguiente repetición.

Para poder observar el progreso de los estudiantes y poder mostrar los valores en tablas, se mostrarán estos valores para la primera y la última respuesta que se hayan realizado. De esta forma, se pueden comparar los resultados iniciales con los más recientes. Por ejemplo: *“Media de la calidad de la primera respuesta a cada pregunta”* o *“Media de la calidad de la última respuesta a cada pregunta”*.

También hay otros valores de interés, como el número de respuestas, el número de días distintos en los que se han realizado respuestas, o el porcentaje de respuestas abiertas con apuntes.

Por otro lado, se podrá estructurar la información a mostrar de diferentes maneras: por cursos, etiquetas, estudiantes y preguntas. También interesa separar los datos de un estudiante en los distintos cursos en los que se encuentre, o distinguir entre etiquetas de distintos cursos. En la tabla A.1 se detallan los valores que se mostrarán para cada conjunto.

Dado que se pretende que los estudiantes vean un conjunto de valores más sencillo sin dar dema-

siados detalles del algoritmo de repaso espaciado, se mostrarán de otra manera en estos casos. Por ejemplo, en vez de mostrar valores de calidad de respuestas o facilidad de las tarjetas, se mostrará el porcentaje de acierto. Además, como sólo podrán ver de manera directa sus propios datos, no será necesario poder agrupar la información por estudiantes. En la tabla A.2 se detallan los datos que verán los estudiantes.

3.3. Análisis de eventos de objetivos

También se quiere mostrar datos de las interacciones de los usuarios con los objetivos. Esta información es menos compleja que la de las respuestas.

Se registran tres tipos de eventos: acceso a objetivos, respuesta de preguntas y subida de archivos. Estos dos últimos tipos de eventos sólo aplicarán para objetivos que incluyan preguntas o que soliciten subir archivos, respectivamente.

Se mostrará una tabla con el número de sucesos de cada tipo, así como un gráfico mostrando la distribución de estos eventos en una línea temporal, de forma similar al gráfico de las respuestas. Dado que las respuestas a preguntas de objetivos quedan registradas también, se podrá mostrar el porcentaje de acierto de las preguntas.

3.4. Filtrado de los datos

Un aspecto fundamental en la realización de este módulo es el filtrado de los datos. El filtrado se podrá realizar de tres maneras: por curso, por etiqueta y por usuario. Será posible, naturalmente, seleccionar más de un elemento de cada tipo o combinar los filtros.

Respecto al filtrado por etiquetas, esto se hace mediante un campo contenido de las preguntas. Para su uso en este módulo, se considerarán distintas las etiquetas para cada curso. Por ejemplo, “variables” en un curso de *Programación en C* y otro de *Programación en Kotlin*, aunque se trata de un concepto parecido, al estar en distintos cursos dispondrán de explicaciones y preguntas distintas, y los contenidos podrán ser accedidos por estudiantes diferentes.

3.4.1. Aplicación de los filtros

Los filtros se aplicarán siempre que sea posible, y no sólo para los datos de la pestaña correspondiente. Por ejemplo, filtrar por un grupo de estudiantes no sólo sirve para mostrar sus datos en la pestaña *Estudiantes*. También se usarán los datos de estos usuarios a la hora de mostrar la facilidad de las respuestas de los cursos en conjunto, el número de preguntas respondidas, los resultados por

etiquetas, etc.

3.4.2. Acceso a los filtros

El administrador dispondrá de un menú en el que poder seleccionar el conjunto de datos por los que filtrar. También podrá guardar una selección de estudiantes (por ejemplo, los estudiantes de un curso en el momento actual) para poder filtrar más rápidamente.

Por otro lado, los estudiantes no pueden controlar de manera directa el uso de los filtros, para simplificar la funcionalidad. En su lugar, cada estudiante verá automáticamente los datos de los cursos a los que está inscrito para sus propios datos.

Si bien queremos que un estudiante no pueda ver los datos específicos de los demás por cuestión de privacidad, sí que se podrán mostrar valores medios de los estudiantes del mismo curso académico. El administrador se encargará de gestionar quiénes forman parte de los grupos.

3.4.3. Enfocar etiquetas y usuarios

También se pretende que, una vez filtrada la información, se pueda "enfocar" los datos de un usuario concreto respecto a los demás. Esto consistirá en destacar visualmente los datos de dicho estudiante para facilitar la comparación con el resto de un curso. Por ejemplo, a la hora de mostrar un histograma, mostrar de otro color la barra a la que pertenece el estudiante seleccionado.

De la misma manera, se podría escoger una etiqueta concreta para poder compararla con los resultados de las demás.

3.5. Estructuración del contenido

Los datos del módulo de estadísticas se mostrarán en una única página, en la que se incluirán varias pestañas. De esta forma, se podrá estructurar la información en los diferentes conjuntos explicados en el apartado anterior (cursos, etiquetas, estudiantes, preguntas y objetivos), sin requerir que un usuario tenga que cargar de nuevo una página cada vez que quiera ver un subconjunto de los datos.

Además, esta estructuración permitirá optimizar el procesamiento. Al entrar en el módulo, se cargará simplemente el "marco" con la cabecera y otra información general, y después, se podrán obtener los datos de cada pestaña de forma independiente. De esta manera, se permite su procesamiento en paralelo, enviando múltiples peticiones al servidor, y reduciendo el tiempo conjunto de carga.

Una ventaja de esto es que si una pestaña requiere menos tiempo de procesamiento, se obtendrá lo antes posible, no quedando ralentizada por las demás. Otra ventaja es que permite que, al cambiar

la selección de filtros, no haga falta recargar la página, sino simplemente actualizar sus contenidos, reduciendo una vez más el tiempo de carga y mejorando la interactividad de la aplicación.

El marco de la página tendrá una cabecera similar a la que hay en los otros módulos: con un botón de cerrar sesión, enlaces a otras páginas de la aplicación, y botones específicos de este módulo. Para esto último, se podrá poner un botón para abrir un menú con los filtros.

Justo debajo de la cabecera, se añadirá un conjunto de botones para seleccionar la pestaña a mostrar, con un aspecto similar al utilizado en las opciones del módulo editor, como se muestra en la figura 3.1.

Figura 3.1: Cabecera y barra de botones del módulo editor

Para cada pestaña, se mostrará una tabla con la información detallada de la información del conjunto correspondiente, histogramas de los datos si procede, y gráficos adicionales que completen la información.

4

Implementación

Una vez que hemos mostrado el diseño del nuevo módulo, sólo queda realizar la implementación. A continuación se explicará cómo se han podido llevar estas ideas a la práctica: el uso de bibliotecas externas, los cambios necesarios en la base de datos, y finalmente los resultados obtenidos.

4.1. Bibliotecas utilizadas

Teniendo en cuenta las necesidades del módulo, se ha decidido usar una serie de bibliotecas para facilitar el desarrollo.

4.1.1. Estilo de los elementos

Se ha utilizado la misma biblioteca que en los módulos ya existentes: Semantic UI [4]. Esta biblioteca dispone de una variedad de funcionalidades que ha simplificado la realización del módulo y además ha permitido mantener una buena apariencia.

La utilidad “dropdown” ha sido la funcionalidad más fundamental, que permite la selección de los filtros a utilizar. Otras utilidades han sido: tabulación, popups, iconos, algunas animaciones, inputs de los formularios...

4.1.2. Gestor de gráficos

En los requisitos de la aplicación estaba el uso de gráficos para mostrar visualmente algunos de los resultados. Se pretenden mostrar gráficos simples de barras para mostrar algunos valores, histogramas para agrupar conjuntos de valores, y diagramas con fechas en el eje de coordenadas, para mostrar valores a lo largo del tiempo.

Se ha usado la biblioteca ChartJS [5] para mostrar los gráficos a partir de los datos. Se muestra un ejemplo de gráfico en la figura 4.1.

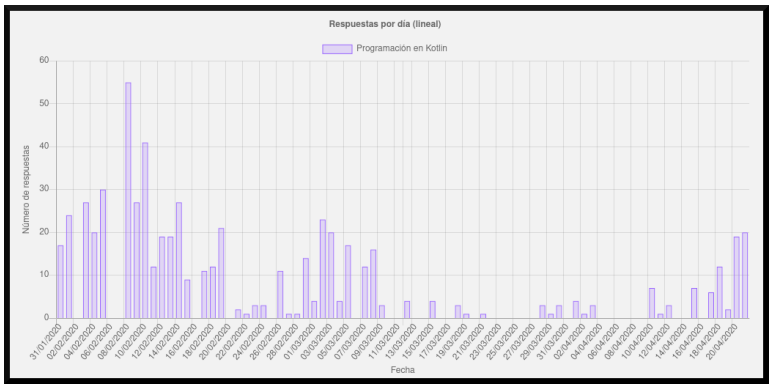


Figura 4.1: Ejemplo de gráfico con ChartJS

Al comenzar el desarrollo, se decidió utilizar dicha biblioteca, dado que disponía de distintos tipos de gráficos así como de un estilo acorde a la aplicación ENCODE. Tras avanzar el desarrollo, se planteó la realización de histogramas para agrupar determinadas series de datos, pero resultaba que esta biblioteca no dispone de dicha opción. Sí que es posible el uso de diagramas de barras, pero dichas barras deben tener unos valores predefinidos antes de la creación del diagrama.

En este punto, se planteó cambiar a la biblioteca Google Charts [6], dado que sí incluía la opción de crear histogramas. Se probó a realizar cambios en todos los gráficos realizados hasta el momento, incluyendo los que no eran histogramas, con el objetivo de unificar utilizar una única librería. Tras realizar el cambio, se ha realizado una comparativa para decidir cuál de las dos bibliotecas se debería utilizar, que se muestra en la tabla 4.1.

	ChartJS	**Google Charts**
Accesibilidad al código	**Se puede guardar** el código en nuestro servidor, y el usuario puede mantener el script correspondiente en su caché. Esto hace que se procesen con **más rapidez** los datos, permite realizar cambios manuales en el código de ser necesario y favorece la depuración de errores en su uso. También se pueden realizar modificaciones en el código de ser necesario.	Para su uso, los datos se envían al servidor correspondiente, y se reciben los datos ya procesados. Esto supone **más lentitud** en la ejecución e impide la personalización.
		Continúa en siguiente página

Tabla 4.1: Comparativa entre funcionalidad de ChartJS y Google Charts

	ChartJS	Google Charts
Creación de histogramas	**No incluye histogramas.** Para mostrar datos de esta manera, se debe crear un diagrama de barras con un código propio para el cálculo del ancho y alto de las barras.	**Posibilidad de crear histogramas.** Permite pasarle el conjunto de datos y genera automáticamente un histograma con un tamaño de barras apropiado. También se puede pedir un tamaño determinado de barras y se dispone de una infinidad de opciones adicionales.
Rendimiento	**Rendimiento adecuado.**	Mal rendimiento. Al probar a realizar los gráficos con esta biblioteca, la página responde con **mayor lentitud** a las interacciones.
Integración en el módulo realizado	**Funcionamiento correcto** para los casos de uso de la aplicación.	**Procesamiento incorrecto en ciertos casos.** La página realizada consta de varias pestañas, una sola activa en cada momento. Resulta que, con esta biblioteca, si se procesa un gráfico que se encuentra en una pestaña no visible, no se dibuja correctamente: la leyenda se sale de los límites del gráfico y los valores mostrados en los ejes se juntan. Se muestra un ejemplo de este resultado en la figura 4.2. Ha habido comentarios al respecto en foros de internet desde hace varios años y todavía no ha sido solucionado [7].

Tabla 4.1: Comparativa entre funcionalidad de ChartJS y Google Charts

Los dos últimos factores que se muestran en la tabla comparativa han sido los más decisivos, dado que afectan de manera directa a la usabilidad de la aplicación. Con esto se tomó la decisión de continuar usando ChartJS. El único inconveniente que esto suponía era que había que realizar

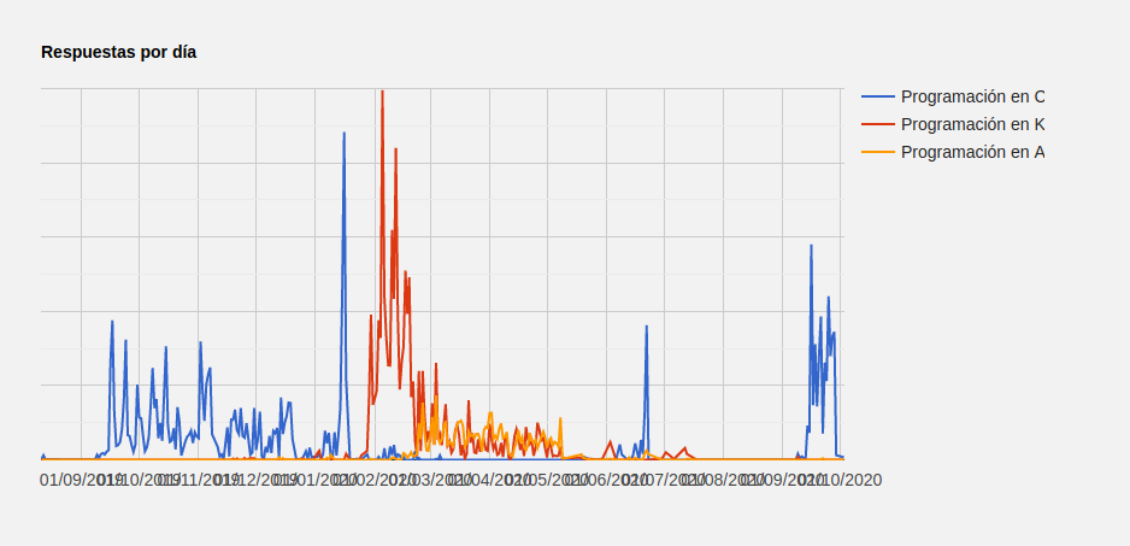


Figura 4.2: Funcionamiento anómalo de Google Charts

una implementación personalizada de histogramas, agrupando los datos de la manera apropiada para generar un gráfico de barras equivalente. Para calcular el tamaño más apropiado para las barras, se ha utilizado la fórmula de Sturges [8].

4.1.3. Gestor de tablas

La mayor parte de la información de la aplicación se muestra en forma de tablas. Para su creación, Django toma los datos procesados del servidor y los introduce en el template, pero se ha querido permitir la reordenación de los elementos de las tablas de forma dinámica mediante la selección de ciertas columnas.

Para permitir esta reordenación, se ha utilizado el script fancyTable.js [9], que permite la gestión de este aspecto. Se han personalizado algunas funciones del código para adaptar su funcionalidad, para permitir por ejemplo la comparación de números, y para permitir cambiar la paginación después de haber sido creada.

4.2. Cambios en base de datos

A continuación se explican los cambios necesarios en la base de datos para poder permitir guardar la información nueva requerida por el módulo o para facilitar su funcionamiento.

4.2.1. Consultas sobre registros de respuestas

Para obtener los datos deseados es necesario realizar consultas en determinadas tablas y aplicando los filtros obtenidos. La clase más importante es *FlashCardHistory*, que contiene los registros de respuestas. Dichos registros disponen de referencias al usuario y a la pregunta, con lo cual se puede realizar el filtrado por usuario, curso y etiqueta. Sin embargo, ha sido necesario crear nuevas tablas para apoyar la realización de estas consultas.

CacheUserTest

Contiene para cada par de *User* y *UnitTest* (equivalente a la clase *FlashCard* del módulo tutorial) una referencia al objeto *FlashCardHistory* más antiguo y otra al más reciente.

Esta clase se ha creado para lidiar con consultas del tipo “media de la facilidad de las primeras/últimas respuestas”. Las consultas necesarias con el esquema de clases inicial resultaban complejas, poco comprensibles, y bastante lentas. Al disponer de esta clase, es posible acceder directamente a los parámetros de los registros primeros y últimos, para un subconjunto de usuarios y preguntas.

Sin esta clase, era necesario realizar las consultas sobre el conjunto de registros, de la clase *FlashCardHistory*. Había que agrupar los datos por *User* y *UnitTest*, obtener las fechas máximas de cada grupo, y después calcular la media del valor buscado. El problema es que en SQL no es posible realizar esas dos operaciones de agregación simultáneamente. Tampoco es posible, al obtener las fechas mínimas/máximas, conseguir también otros valores como la facilidad o incluso el identificador de las filas, puesto que esas columnas no forman parte del agrupador. La única solución (sin crear una clase auxiliar) sería realizar una consulta adicional filtrando por los valores de *User*, *UnitTest*, y fecha mínima/máxima, y entonces calcular la media del valor buscado.

Para evitar tener que realizar consultas tan complejas que afecten el rendimiento de la aplicación, se ha creado la clase *CacheUserTest*, que permite guardar los valores de respuestas más recientes y antiguos. Esto permite hacer las consultas deseadas de forma trivial. Se ha hecho un código personalizado de migración para generar objetos *CacheUserTest* a partir de los objetos *FlashCardHistory* existentes, y en el módulo tutorial, se ha añadido código que asegura la creación y actualización de estos objetos cada vez que se crea o actualiza un *FlashCardHistory*.

Al crear esta clase, en la aplicación había aproximadamente 80000 registros para *FlashCardHistory*, y con ello se creaban unas 20000 filas para *CacheUserTest*. La creación inicial de las filas de dicha tabla requería unos 90 segundos, que es un tiempo despreciable, dado que sólo se tenía que realizar una vez. Por otro lado, la creación o actualización de estos objetos durante el uso normal de la aplicación no requiere apenas tiempo y por tanto no afecta al rendimiento.

Se ha realizado un script de test que compara los tiempos de ejecución al hacer las consultas directamente con *FlashCardHistory*, y la optimización que supone el uso de *CacheUserTest*. En el

segundo caso, el tiempo de consultas se reduce en aproximadamente un 85 %, por lo que se ha conseguido el resultado esperado.

Hay que tener en cuenta que esta clase no facilita el filtrado por fecha (dado que hasta ahora no ha sido una necesidad), sino que contiene sólo la primera y la última referencia disponible. Si se quisiera filtrar por fecha, habría que considerar otro tipo de solución más complejo.

CourseTag

Los objetos de la clase *UnitTest* pertenecen a un curso determinado, y tienen una etiqueta que permite clasificar la pregunta. Dado que se pretende distinguir las etiquetas de distintos cursos, es necesario poder filtrar las preguntas por estos pares de valores, pero esto no se puede hacer en SQL salvo que se conozcan los valores de antemano, lo cual no es posible para lo que se quiere realizar.

Por ello se ha creado esta clase, que representa un par curso-etiqueta. Se ha añadido en *UnitTest* una referencia a un objeto *CourseTag*, para disponer de un campo que represente estos dos valores, y así, las consultas se pueden realizar fácilmente.

Otra ventaja es que permite que se puedan poner en un select de html, utilizando como valores los identificadores de la nueva entidad. Si se realizara con el propio texto de la etiqueta, podría haber problemas si alguna etiqueta contuviera una coma, porque los valores escogidos se guardan separados por comas.

Se ha realizado de una manera que no ha requerido cambiar el código previo que se encargaba de gestionar la creación de tests: las etiquetas de las preguntas (*UnitTest*) siguen siendo un campo de texto. A la hora de guardar un UnitTest (tanto para crear uno nuevo como para cambiar datos), se comprueba la etiqueta y el curso y se asocia un *CourseTag* (creando uno nuevo de ser necesario).

Se ha creado un código personalizado de migración para generar objetos *CourseTag* a partir de los UnitTest existentes y guardar las referencias.

4.2.2. Creación de clases adicionales

Se han creado otras clases para complementar la funcionalidad del nuevo módulo.

StudentGroup

Se trata de una lista de usuarios, una lista de cursos, y un nombre. Esto permite que el administrador pueda filtrar fácilmente por estudiantes que estén cursando una determinada asignatura. También dispone de un campo que determina si el grupo es *público*. En caso de serlo, los estudiantes pertenecientes a él pueden ver los datos medios del grupo cuando acceden a sus estadísticas. De esta forma, el administrador puede decidir si un estudiante puede ver información del conjunto de su clase.

Aunque el entorno Django proporciona una clase *Group* (al igual que una clase *User*), su funcionalidad está orientada a los permisos que tienen los usuarios a la hora de user la interfaz de administración, en caso de que hubiera múltiples usuarios con acceso. Para este caso particular no tendría utilidad porque sólo un usuario tiene acceso. Como además se querían asociar campos específicos de esta aplicación, es mejor crear una clase personalizada para esta funcionalidad.

UserExtraData

Se trata de una clase para extender los atributos de la clase *User*, proporcionada por Django. Por el momento sólo contiene la referencia al usuario correspondiente y un booleano que dice si el usuario es de prueba. Esto se ha hecho para que el administrador pueda filtrar los usuarios que no resultan de interés para el módulo de estadísticas, y que no salga de forma predeterminada en el menú de selección. Esta clase podría ampliarse en el futuro para almacenar preferencias de los usuarios.

Se ha personalizado la interfaz de administración para la clase *User*, mostrando si los usuarios son de prueba, y permitiendo seleccionar varios estudiantes y realizar la acción de establecerlos de prueba o no.

4.3. Cambios en módulos previos

Para asegurar la integración del nuevo módulo con el resto de la aplicación, se han realizado unos cambios en los módulos ya existentes para que los usuarios habituales puedan acceder a la nueva funcionalidad.

4.3.1. Enlaces al módulo de estadísticas

Tanto en el módulo tutorial como en el editor, en la barra superior se ha añadido un enlace que lleva al módulo nuevo.

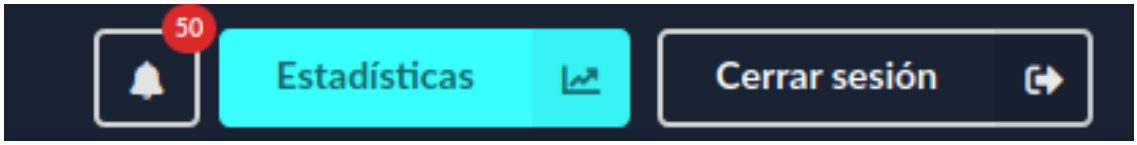


Figura 4.3: Enlace para acceder al módulo de estadísticas desde el módulo de tutorial

Además, al terminar las preguntas diarias, se redirige al usuario al módulo charts, en lugar de volver a la página de cursos. El usuario puede volver al módulo principal pulsando el botón “Tutorial” en la barra superior.

Al completar una unidad y realizar sus preguntas, también se redirige al módulo de estadísticas, en

este caso filtrando por el curso que estuviera haciendo. En la parte superior habrá un botón “Volver al curso”, que le lleva al menú de unidades del curso que estaba haciendo, como haría normalmente.

Figura 4.4: Enlace para volver al curso desde el módulo de estadísticas

4.3.2. Descargas

Como se ha mencionado, el módulo editor incluía una opción para descargar los objetos de la clase *FlashCardHistory* de todos los usuarios, para poder realizar un análisis mediante otra aplicación. Se ha decidido mantener esta funcionalidad, por si se quisiera obtener esta información, y se ha ampliado para mejorar su utilidad.

Primero, se ha movido al módulo de estadísticas, que hace que su uso sea más coherente. Se accede seleccionando el menú de opciones en la barra superior. Una vez en el menú de descargas, se seleccionan los datos a descargar. Hay una opción para la descarga de *FlashCardHistories* (como era en un principio), opciones para descargar las tablas que se muestran en el propio módulo (cursos, etiquetas, estudiantes, preguntas, objetivos), y una opción para descargar los datos que se muestran en el gráfico de respuestas diarias.

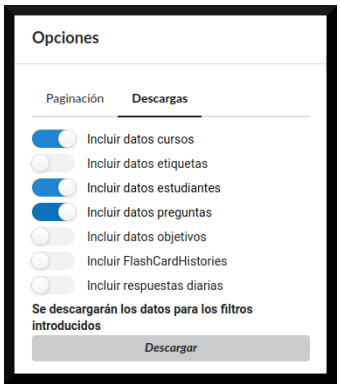


Figura 4.5: Menú de descargas

Si se selecciona una sola opción, se descargará como archivo csv; en caso de escoger más de una, se obtendrá un archivo zip con los archivos csv correspondientes.

Además, todos los datos se filtran por la selección de usuarios, cursos y etiquetas que se hayan escogido en la pestaña de filtros. De esta manera, se pueden obtener datos de *FlashCardHistory* de un subconjunto de los datos, lo cual no se podía hacer antes.

4.4. Apariencia

Cuando se accede al módulo de estadísticas, se carga la parte superior de la página, y se realizan varias peticiones asíncronas, mediante AJAX, para obtener las pestañas con los datos.

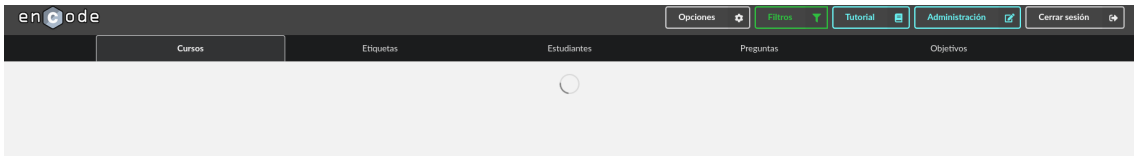


Figura 4.6: Pestaña de estadísticas cargando

Todas las pestañas, con la excepción de la pestaña de cursos, separan la información de los distintos cursos. En la parte superior se muestra el curso actual, así como unos botones que permiten cambiar la selección (en el caso de que haya varios cursos disponibles), mostrando así diferentes tablas y gráficos, como se muestra en la figura 4.7.

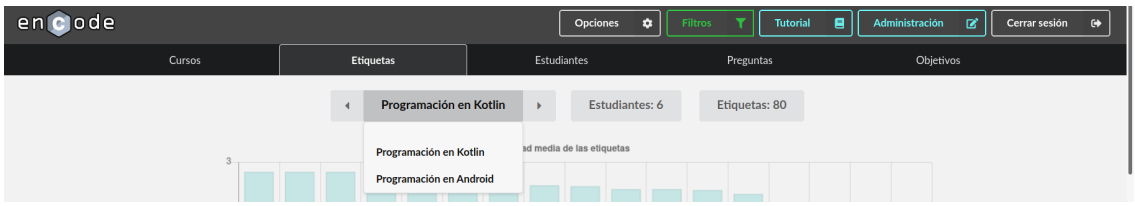


Figura 4.7: Selección de curso en estadísticas

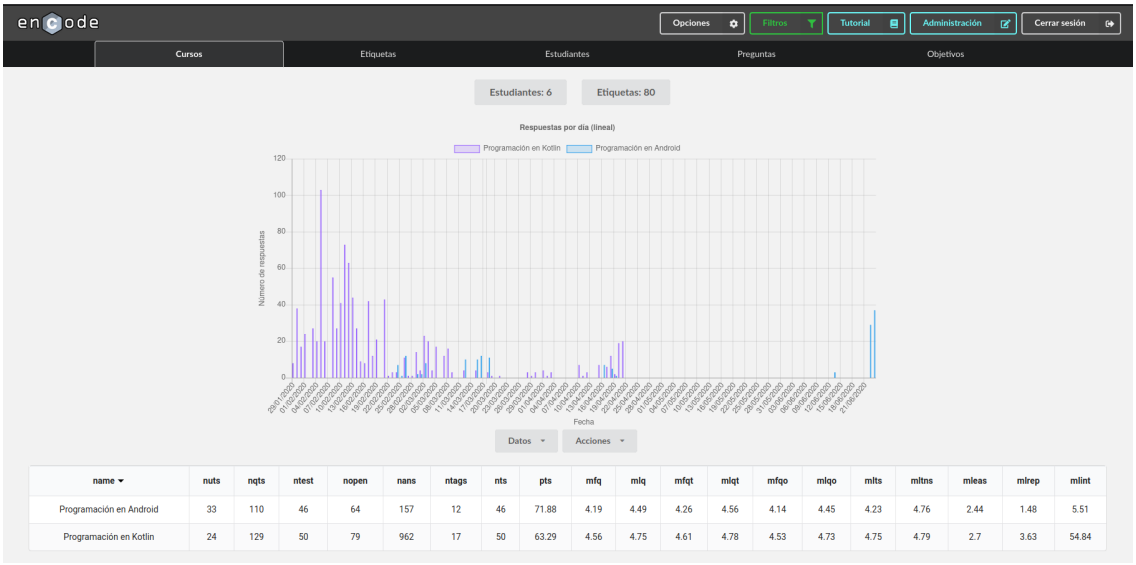
Pestaña de cursos

Muestra datos generales de los cursos: un gráfico que indica el número de respuestas según el tiempo y una tabla de información.

El gráfico muestra distintas barras para los diferentes cursos, aplicando los filtros de etiquetas y usuarios. Los cursos pueden ocultarse o mostrarse pulsando los iconos de las leyendas en la parte superior. Mediante el botón *Datos* en la parte inferior del gráfico, se permite alternar entre respuestas diarias, por semana, por mes, por hora del día y por día de la semana. También se puede mostrar la escala de forma lineal o logarítmica.

Además, pinchando con el ratón en la gráfica y arrastrándolo horizontalmente, se puede realizar zoom para ver de forma ampliada. Esto se puede devolver a la normalidad pulsando una de las opciones disponibles mediante el botón *Acciones*.

El gráfico también permite ocultar valores atípicos, útil para ocultar fechas con pocas respuestas y que estén muy separadas de los datos principales. Es posible mostrarlos u ocultarlos pulsando otro botón disponible en *Acciones*. Más detalles sobre la gestión de valores atípicos en el anexo B.3.



name	nuts	nqts	ntest	nopen	nans	ntags	nts	pts	mfq	mlq	mfqt	mlqt	mfqo	mlqo	mlts	mltns	mleas	mlrep	mlint
Programación en Android	33	110	46	64	157	12	46	71.88	4.19	4.49	4.26	4.56	4.14	4.45	4.23	4.76	2.44	1.48	5.51
Programación en Kotlin	24	129	50	79	962	17	50	63.29	4.56	4.75	4.61	4.78	4.53	4.73	4.75	4.79	2.7	3.63	54.84

Figura 4.8: Pestaña de cursos

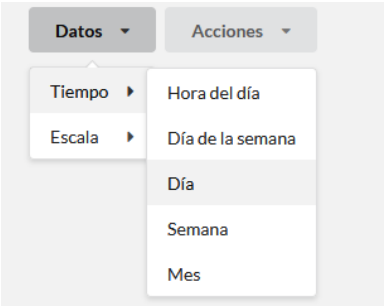
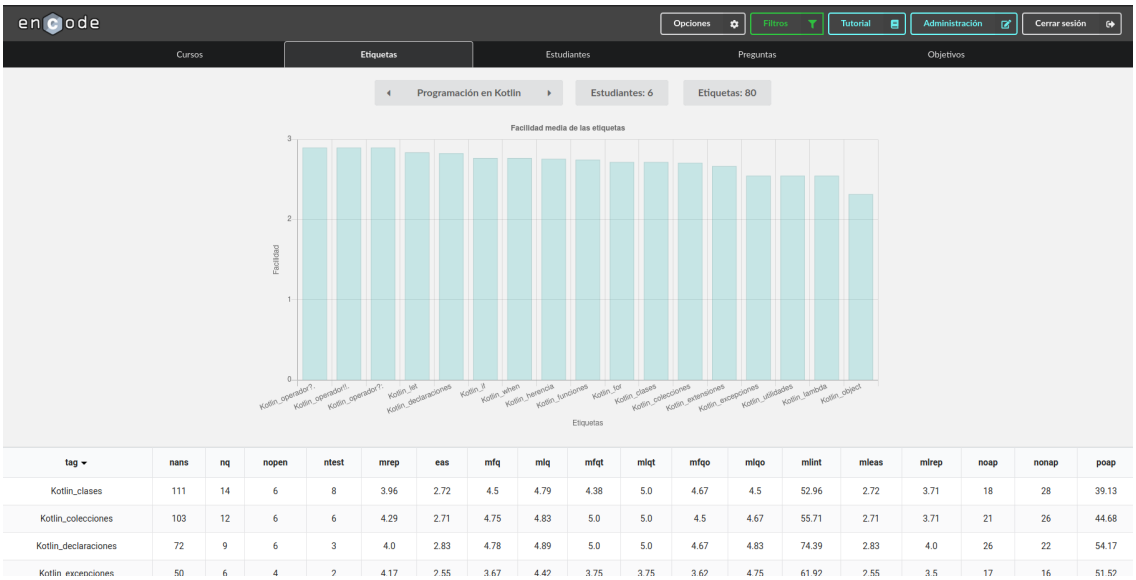


Figura 4.9: Opciones de gráfico de respuestas

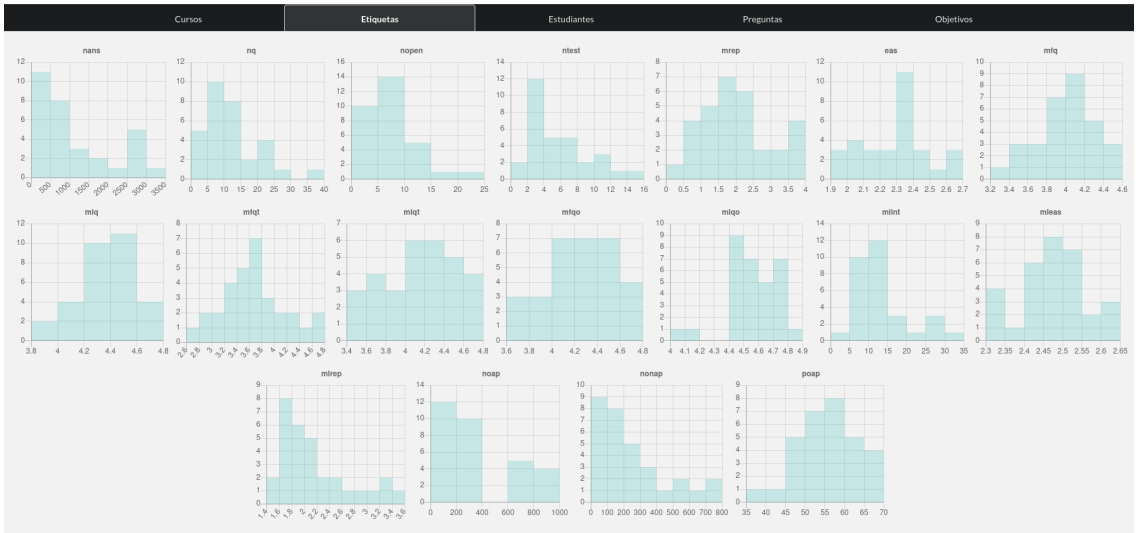
Pestaña de etiquetas

Muestra, para cada curso, una tabla con los datos de sus etiquetas, una serie de histogramas para la mayoría de sus parámetros numéricos, y un diagrama de barras para representar el valor *facilidad*. Se muestran imágenes en la figura 4.10.



tag	nans	nq	nopen	ntest	mrep	eas	mfq	mlq	mfqt	mlqt	mfqo	mlqo	mlint	mleas	mlrep	noap	nonap	poap
Kotlin_clases	111	14	6	8	3.96	2.72	4.5	4.79	4.38	5.0	4.67	4.5	52.96	2.72	3.71	18	28	39.13
Kotlin_colecciones	103	12	6	6	4.29	2.71	4.75	4.83	5.0	5.0	4.5	4.67	55.71	2.71	3.71	21	26	44.68
Kotlin_declaraciones	72	9	6	3	4.0	2.83	4.78	4.89	5.0	5.0	4.67	4.83	74.39	2.83	4.0	26	22	54.17
Kotlin_excepciones	50	6	4	2	4.17	2.55	3.67	4.42	3.75	3.75	3.62	4.75	61.92	2.55	3.5	17	16	51.52

(a) Gráfico principal y tabla de etiquetas



(b) Histogramas de etiquetas

Figura 4.10: Pestaña de etiquetas

Pulsando en las barras de los histogramas, se puede ver las etiquetas correspondientes a los rangos de valores, junto con los valores particulares, como se muestra en la figura 4.11. Se incluye un botón para mostrar u ocultar dichos valores, y otro botón para copiar la información al portapapeles.

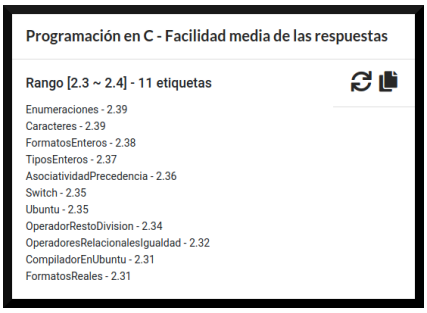


Figura 4.11: Detalles de histogramas en etiquetas

Pestaña de estudiantes

Para cada curso, muestra una tabla que tiene una fila por cada estudiante escogido que esté matriculado en ese curso, así como unos histogramas agrupando los datos de la mayoría de las columnas de dicha tabla. También se incluye un gráfico que muestra cuántos estudiantes hay en las unidades. Se muestran imágenes en la figura 4.12.

Cuando se pulsa en las barras de los histogramas, se puede ver qué estudiantes se corresponden a los datos, de forma similar a como funciona en la pestaña de etiquetas. También se puede hacer al pulsar en las barras del gráfico de unidades, para ver qué estudiantes están en una cierta unidad, como se visualiza en la figura 4.13.

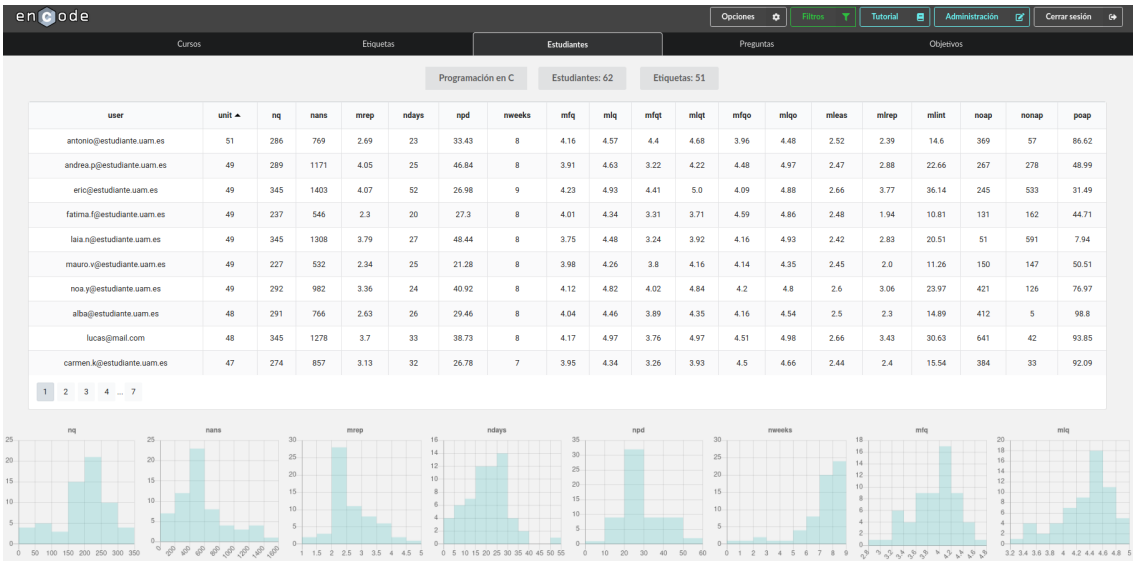
Pestaña de preguntas

Muestra, para cada curso, una tabla con el texto de las preguntas y datos asociados a las mismas. También se muestra un histograma con los valores de los intervalos de las tarjetas. Se alternar entre escala lineal o logarítmica, pulsando el botón en la parte inferior del gráfico. En la figura 4.14 se muestra la apariencia de esta pestaña.

Pestaña de objetivos

Muestra una tabla por curso con el número de accesos a objetivos, archivos subidos, y respuestas a preguntas de objetivos. También se muestra el porcentaje de acierto de las preguntas respondidas. Se pueden mostrar los datos de objetivos individuales o de unidades en conjunto. Se muestran ejemplos de ambas formas de visualizar la tabla en la figura 4.15.

También se muestra un gráfico similar al de la pestaña de cursos, en este caso con la distribución de los eventos a lo largo del tiempo, como se ve en la figura 4.16. Incluye las mismas opciones: se puede alternar entre distintas frecuencias de tiempo, escalas, funcionalidad de zoom y ocultar valores atípicos. Se incluye una opción adicional para alternar entre los distintos tipos de eventos, como se muestra en la figura 4.17.



(a) Tabla e histogramas de estudiantes

(b) Gráfico de progreso en unidades

Figura 4.12: Pestaña de estudiantes

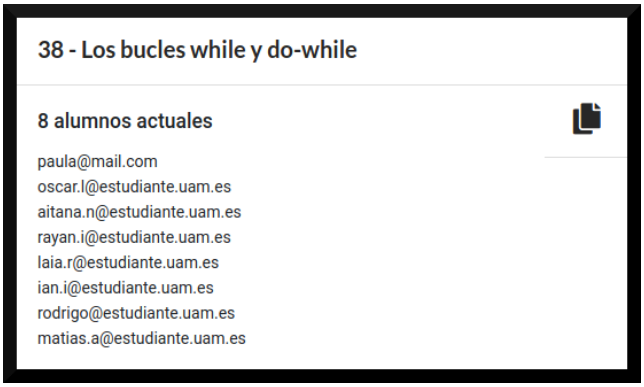


Figura 4.13: Lista de estudiantes en una cierta unidad

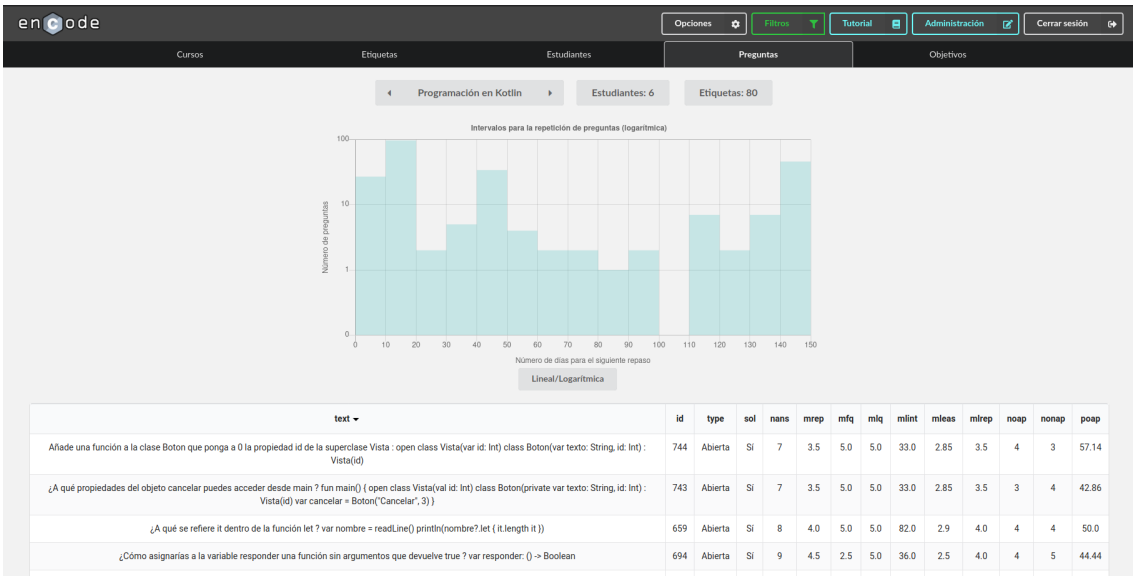


Figura 4.14: Pestaña de preguntas

4.4.1. Pestañas para estudiantes

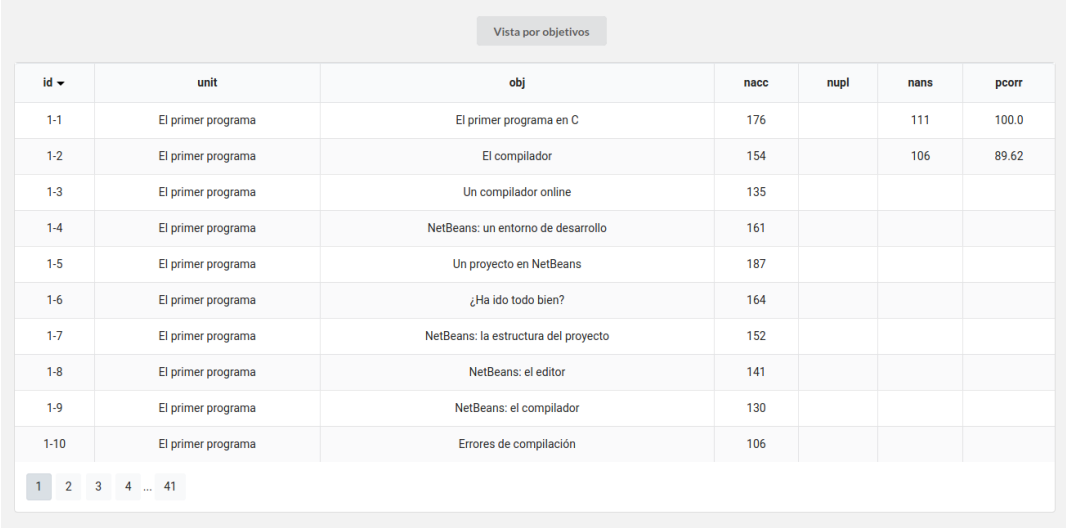
Los estudiantes sólo pueden ver sus propios datos, de modo que no tiene sentido que tengan acceso a la pestaña *Estudiantes*. En su lugar, en la pestaña de *Cursos*, se muestra información sobre su actividad en cada curso en el que estén matriculados.

Respecto a la información mostrada en las distintas tablas, en lugar de mostrar parámetros característicos de SM2 [2], se muestra el porcentaje de acierto de las primeras y últimas respuestas. Esto se puede ver en las tablas de *Cursos* y *Etiquetas*, y para *Preguntas*, se muestra si la primera y la última respuesta se han acertado (dado que no tiene sentido calcular una media para preguntas individuales). Además, el gráfico de la pestaña *Etiquetas* muestra el porcentaje de acierto de las últimas respuestas para cada etiqueta.

La pestaña *Objetivos* es la única que no tiene diferencias con la versión del administrador, simplemente muestra la información de los eventos para los datos propios del estudiante.

Si un estudiante pertenece a un grupo *público*, se muestran los valores medios del grupo en una tabla separada, dentro de la pestaña de *Cursos*. También se puede ver que en el gráfico se muestra el número medio de respuestas del grupo para la unidad de tiempo correspondiente en forma de línea. Se muestra un ejemplo de esta vista en la figura 4.18.

Los datos de grupo se muestran de manera que no se puede saber los datos específicos de un estudiante, sino que se calculan los datos medios en el servidor y a los usuarios les llega la información ya procesada. Además, el administrador tiene el control sobre los grupos de estudiantes que pueden ver los datos medios, lo cual se explica en el apartado 4.5, de modo que puede controlar que se trate de un conjunto del que no resulte posible recrear los datos originales. Por ejemplo, para un grupo de



id	unit	obj	nacc	nupl	nans	pcorr
1-1	El primer programa	El primer programa en C	176		111	100.0
1-2	El primer programa	El compilador	154		106	89.62
1-3	El primer programa	Un compilador online	135			
1-4	El primer programa	NetBeans: un entorno de desarrollo	161			
1-5	El primer programa	Un proyecto en NetBeans	187			
1-6	El primer programa	¿Ha ido todo bien?	164			
1-7	El primer programa	NetBeans: la estructura del proyecto	152			
1-8	El primer programa	NetBeans: el editor	141			
1-9	El primer programa	NetBeans: el compilador	130			
1-10	El primer programa	Errores de compilación	106			

1 2 3 4 ... 41

(a) Tabla de objetivos

id	unit	nacc	nupl	nans	pcorr
1	El primer programa	1614		217	94.93
2	El compilador	1061		569	77.15
3	Tipos de datos y variables	1008		519	82.85
4	Las funciones printf y scanf	1253	120	394	66.75
5	Los caracteres	846	61	142	89.44
6	Ponte a prueba 1	755	319	130	95.38
7	Bienvenido a Ubuntu	1784		400	77.0
8	El compilador en Ubuntu	1593		416	76.44
9	Compilar y enlazar en Ubuntu	929	62	328	79.88
10	La instrucción if	689	186	66	81.82

1 2 3 4 ... 6

(b) Tabla de objetivos agrupada por unidades

Figura 4.15: Tablas de pestaña de objetivos

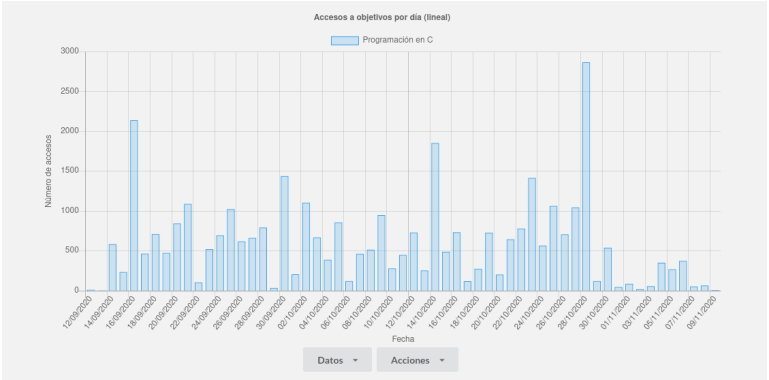


Figura 4.16: Gráfico de objetivos

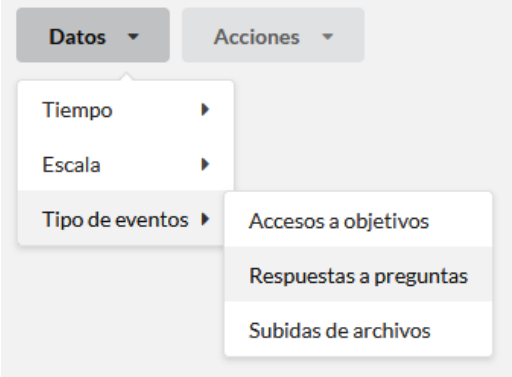


Figura 4.17: Opciones de gráfico de objetivos

dos estudiantes, sería trivial para un estudiante averiguar los datos del otro, pero para grupos de más 30 personas, no resulta posible sacar ninguna conclusión de los datos reales a partir de unos datos medios.

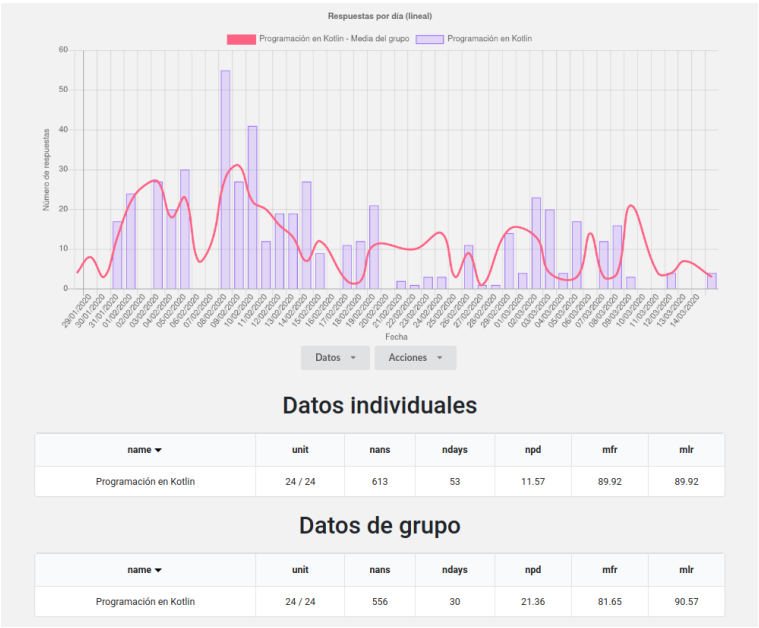


Figura 4.18: Vista de datos de grupo para un estudiante

4.5. Filtros

El menú de filtros, al que se accede pulsando un botón en la barra superior sólo para el administrador, está formado por varias pestañas con diferentes opciones.

Usuarios

La pestaña de usuarios permite escoger los usuarios por los que filtrar. Incluye una opción para seleccionar los usuarios mediante un archivo de texto, que incluya los nombres de los usuarios (en forma de correo electrónico). Esta opción es útil para los administradores porque pueden disponer de listados de estudiantes de un curso, y con ello realizar la selección de estos estudiantes fácilmente.

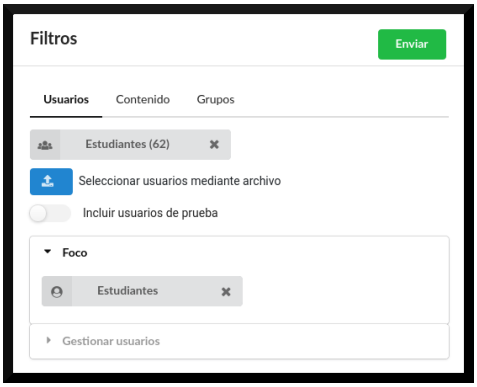


Figura 4.19: Opciones de filtros de usuarios

También se permite seleccionar unos estudiantes sobre los que "enfocar", de modo que se destacan sus datos en las gráficas, como se muestra en la figura 4.20.

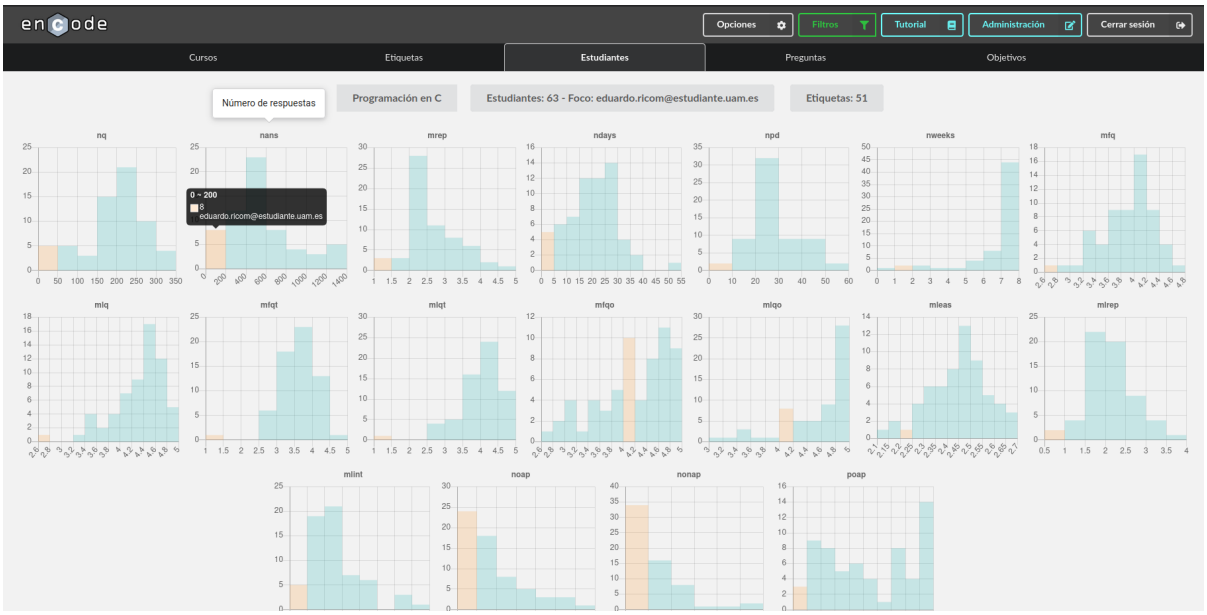


Figura 4.20: Enfocar estudiantes

La opción *Mostrar vista de estudiante* sólo está disponible cuando hay un solo estudiante seleccionado, como se ve en la figura 4.21. Esta opción permite al administrador ver la vista simplificada a la que el estudiante correspondiente tendría acceso.

Figura 4.21: Opción de vista de estudiante

Contenido

La pestaña de contenido permite escoger los cursos y etiquetas que mostrar, además de la posibilidad de utilizar la funcionalidad de foco sobre una selección de etiquetas, como se ve en la figura 4.23.

Figura 4.22: Opciones de filtros de contenido

La opción *Incluir preguntas no respondidas* sirve para las pestañas de *Preguntas* y *Etiquetas*. Dado que los estudiantes requieren un tiempo para avanzar sobre las unidades y responder las preguntas disponibles, de forma predeterminada se excluyen las preguntas no respondidas, y las etiquetas de las que no se haya respondido ninguna pregunta. Sin embargo, es posible utilizar esta opción para incluir todos los datos disponibles, y así poder ver qué preguntas y etiquetas no se han respondido todavía.

Grupos

Por último, está la pestaña de grupos, que se ilustra en la figura 4.24. Esto permite al administrador dejar una selección de estudiantes y cursos, asociados a un nombre, para poder volver a seleccionarse posteriormente. La información de un grupo se puede cambiar tras ser creado.

Pulsando el botón *Nuevo* se añade un bloque para la edición de un nuevo grupo. Una vez creado y editado, pulsando *Seleccionar*, se aplican los filtros del grupo y se recargan las pestañas de contenido.

Figura 4.23: Enfocar etiquetas

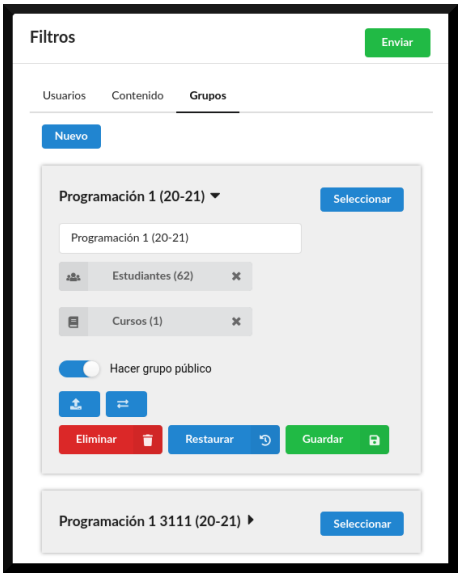


Figura 4.24: Opciones de grupos para filtros

Se dispone de las siguientes opciones de edición de los grupos:

- Guardar

  Envía los datos al servidor para que queden guardados.

- Restaurar

  Descarta los cambios realizados que no hayan sido guardados, de modo que vuelve a mostrar los datos que hubiera la última vez que se guardó.

- Eliminar

  Elimina el grupo, tras mostrar un aviso y solicitar confirmación. Esta acción no se puede deshacer.

- Hacer grupo público

  Si se hace *público* un grupo, los estudiantes pertenecientes podrán ver la media de los datos, como se explica en el apartado 4.4.1.

- Seleccionar mediante archivo

  Funciona igual que en la pestaña de *Usuarios*, pero la selección se aplica al grupo que se esté editando.

- Obtener selección de las otras pestañas

  Selecciona, para el grupo que se esté editando, la selección de usuarios en la pestaña *Usuarios* y los cursos escogidos en la pestaña *Contenido*. De esta forma, si ya se ha hecho una selección en las otras pestañas, no es necesario realizarla de nuevo, sino que esto permite copiarlo automáticamente.

5 Integración, pruebas y resultados

5.1. Integración

Al igual que la versión inicial del proyecto ENCODE, la realización de este módulo se desarrolló de forma local, en base al código fuente previo y la base de datos disponible.

En lo que se refiere a la integración de la funcionalidad nueva con la ya existente, se realizaron algunos cambios en los módulos preexistentes, lo cual se ha explicado en el apartado 4.3.

Cuando el proyecto estaba terminado, se procedió a integrar los cambios en el servidor de ENCODE. Esto se realizó una semana antes del comienzo de las clases del curso 20-21, para poder probar su funcionamiento y poder corregir a tiempo los errores que aparecieran.

La integración fue sencilla, puesto que la aplicación ya estaba configurada correctamente en el servidor, y el resultado de este proyecto sólo suponía extender la funcionalidad ya disponible. La configuración ya existente no tuvo que ser cambiada, salvo por la actualización de la versión de Django, que se explica en el anexo B.2.

Hubo que realizar algunas rectificaciones en el código del módulo nuevo, sobre todo respecto a las migraciones necesarias de la base de datos, que no se ejecutaban de la manera esperada. Una vez realizados estos cambios, se pudo probar correctamente la aplicación.

5.2. Pruebas y mantenimiento

Durante la primera semana, se realizaron pruebas de la funcionalidad de la aplicación y se corrigieron los errores que se encontraron. Posteriormente, con el comienzo del curso, se utilizó la aplicación de manera práctica, y con ello se encontraron algunos errores más y surgieron propuestas de mejora de la funcionalidad.

A continuación se muestran los aspectos corregidos o mejorados junto con la fecha en la que se realizaron.

- **8/9/2020**: Corregido orden de las columnas de las tablas.
- **9/9/2020**
  - Cambiadas abreviaturas de algunos nombres de columnas, para que sean más cortas.
  - Establecidos nombres de los ejes en diagramas en los que faltaban.
  - Corregido error de ordenación de los valores en el diagrama de facilidad de etiquetas.
  - Mostrar, para las gráficas que permiten alternar entre escala lineal y logarítmica, el tipo de escala actual en el título de la gráfica.
  - Reducida la anchura de las barras en el histograma de los intervalos de las tarjetas, para mostrar datos más específicos.
  - Corregida funcionalidad de foco de etiquetas.
- **12/9/2020**: Aparecen nombres de usuarios duplicados. Se debe a que los nombres se estaban mostrando en minúsculas, pero hay casos de usuarios distintos que se distinguen según si el nombre de usuario usa minúsculas o mayúsculas.
- **29/9/2020**: Aparecen etiquetas con parámetro de *facilidad* no nulo pero no habiendo respuestas. Esto se debía a usuarios que tienen tarjetas que no habían respondido todavía.
- **1/10/2020**: Añadida paginación opcional de las tablas, y opción de elegir el valor de paginación.
- **3/10/2020**
  - Permitir que los grupos de estudiantes incluyan una selección de cursos. Esto se debe a que algunos estudiantes pueden estar suscritos en varios cursos, pero el administrador normalmente quiere filtrar por un curso concreto para los estudiantes de una asignatura.
  - Permitir la edición de los grupos ya existentes.
  - Opción de grupos públicos.
  - Mostrar información a los estudiantes sobre los grupos públicos a los que pertenecen.
- **6/10/2020**
  - Mejorado estilo de botones para selección de estudiantes.
  - Mostrar ventana separada para seleccionar usuarios mediante archivo.
- **8/10/2020**
  - Corregido error al ordenar tablas, debido al cambio de paginación.
  - Corregido error al mostrar estadísticas de grupo a un estudiante.
- **20/10/2020**: Mostrar porcentaje de acierto a los estudiantes en vez de la facilidad de tarjetas.
- **23/10/2020**: Acceso opcional a la vista de estudiante para el administrador.
- **24/10/2020**
  - Opción de quitar valores atípicos en gráfico de respuestas por tiempo.
  - Separados los valores individuales y de grupo en tablas separadas para los estudiantes.
- **25/10/2020**: Mostrar en gráfico de respuestas el número medio de respuestas del grupo, para estudiantes pertenecientes a un grupo público.
- **29/10/2020**
  - Mostrar selección actual de usuarios y etiquetas sin tener que entrar al menú de filtros.
  - Mejorado estilo de selección de curso a mostrar en las pestañas.
- **1/11/2020**: Creada pestaña de objetivos.

5.3. Resultados

Se ha realizado una encuesta a los estudiantes del curso que estaban probando el módulo de estadísticas por primera vez. Dado que también era su primera vez utilizando ENCODE, se les ha preguntado tanto de la aplicación en general como del módulo de estadísticas.

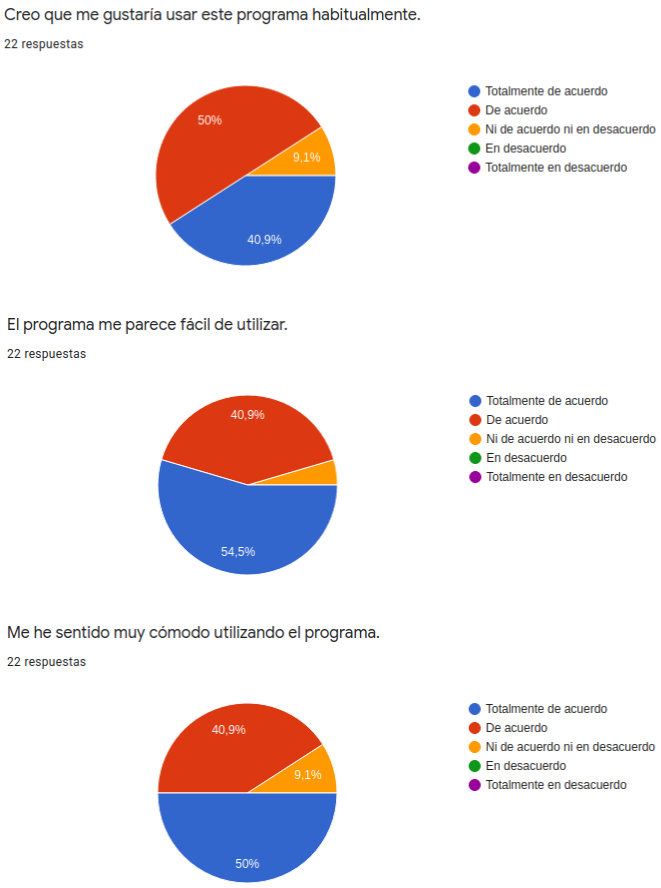


Figura 5.1: Resultados encuesta: valoración de la aplicación

Se puede observar en la figura 5.1 que la mayoría de los usuarios está satisfecho con la funcionalidad de la aplicación y su facilidad de uso. La mayoría afirma que les gustaría poder utilizar esta aplicación habitualmente, por lo que sería interesante que se pudiese utilizar en otras asignaturas.

Por otro lado, respecto a la valoración de algunos aspectos del módulo de estadísticas, hay resultados un tanto diferentes, como se ve en la figura 5.2. Respecto a la utilidad de los parámetros mostrados en las estadísticas, la mitad de los estudiantes están de acuerdo o muy de acuerdo en su utilidad, mientras que la otra mitad no se inclina ni a favor ni en contra.

En cuanto al hecho de ver los datos medios del grupo, la mayoría de usuarios lo valoran de forma positiva, pero aparecen unos pocos usuarios que están en desacuerdo. Esto es comprensible porque se está utilizando la información generada por los usuarios, y es posible que a algunos les preocupe

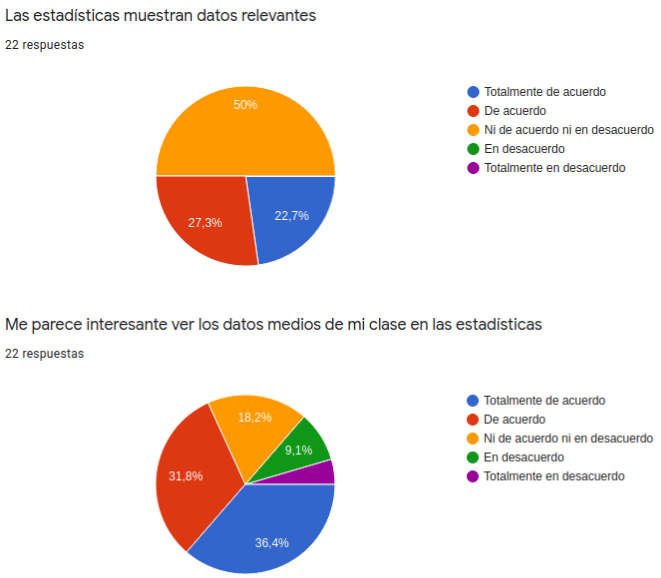


Figura 5.2: Resultados encuesta: valoración del módulo de estadísticas

su privacidad. Dicho esto, los datos de grupo se muestran de manera que no es posible recrear la información original, como se ha explicado en 4.4.1, por lo que no es un aspecto preocupante.

6

Conclusiones y trabajo futuro

6.1. Conclusiones

Gracias a este trabajo, el administrador puede sacar un mayor provecho de la información del progreso de los usuarios. Esto permite saber qué aspectos resultan más difíciles, tanto a nivel individual como conjunto, y comparar los datos de un estudiante con los del resto de su clase, para poder guiarles mejor a la hora de realizar tutorías individuales.

Los estudiantes ahora pueden ver los resultados de sus interacciones, de modo que pueden comprender mejor la utilidad de la aplicación, y con ello animarles a sacar mayor provecho de ella. Teniendo en cuenta las opiniones de los estudiantes encuestados, se podría reconsiderar qué datos se muestran y la manera en la que se hace.

Este trabajo ha cumplido con el propósito de poder permitir realizar un seguimiento de los usuarios de la plataforma, se le ha sacado partido desde su despliegue, y va a seguir siendo útil en el futuro. Se ha realizado de manera que se facilita su escalabilidad, como se detalla en el anexo B.1, por lo que se podrá ampliar fácilmente su funcionalidad en caso de ser necesario.

6.2. Trabajo futuro

Aun habiendo terminado este trabajo, siempre es posible su ampliación para poder aumentar su utilidad o adaptarse a situaciones futuras. Al fin y al cabo, el proyecto ENCODE sigue en desarrollo y todavía se usa en pocas asignaturas.

A continuación se recopilan una serie de sugerencias, algunas sobre el módulo de estadísticas, otras sobre la aplicación en general, sobre aspectos que se podrían mejorar o reconsiderar.

6.2.1. Consultas del módulo de estadísticas

Una buena parte de las consultas del módulo de estadísticas se han planteado para la obtención de parámetros en la primera y la última respuesta, para cada par de usuario y pregunta. Esto permite una comparación general del progreso que ha tenido el estudiante, pero no se dispone de información de los valores intermedios. Se podría intentar plantear estudiar la forma en la que los parámetros han ido evolucionando.

En relación con lo anterior, al mostrar los resultados de la primera y última respuesta, convendría hacer que fuera en un rango de fechas determinado. Esto supondría replantear la manera en la que se gestiona la caché de respuestas, ya que ahora mismo no es flexible en cuanto a cuál es la primera y la última marca de tiempo. Esto serviría para poder mostrar los valores de últimas respuestas para fechas anteriores, y con ello poder mostrar la evolución del rendimiento de los estudiantes.

6.2.2. Gestión de usuarios

Ahora mismo sólo hay un usuario administrador, que puede ver detalles de los usuarios en el módulo de estadísticas. Sería útil permitir a otros profesores, especialmente aquellos involucrados en las asignaturas en las que se usa ENCODE, que pudieran tener un usuario en el que poder visualizar estos datos. Habría que estudiar qué permisos deberían tener dichos usuarios para poder ver la información, pero limitando algunas funcionalidades, como la creación de grupos.

Por otro lado, cualquier persona puede crear cuentas ilimitadas en la aplicación. Se podría hacer uso de un captcha para evitar el spam, que podría ser un problema en el futuro, y también el envío de correos de autenticación, para validar que las cuentas son reales. También habría que plantear si la aplicación es solo accesible para usuarios de la UAM o incluso que estén matriculados en ciertas asignaturas, y buscar una manera de verificar este aspecto.

6.2.3. Otros aspectos a considerar

La funcionalidad del algoritmo de repaso espaciado se podría ampliar. Por ejemplo, la aplicación Anki [3], que implementa este algoritmo, hace que las preguntas realizadas por primera vez en un mismo día no se repitan en un mismo día posterior, y así se repartan en días cercanos. Esto sería apropiado para ENCODE, dado que cada vez que el estudiante termina una unidad, se añaden múltiples preguntas a la vez a su lista de tarjetas.

Bibliografía

[1] A. Frías Díaz, "Encode: Tutor online de programación," bachelor's thesis, Universidad Autónoma de Madrid, 5 2019.
URL: http://hdl.handle.net/10486/688876.

[2] P. Woźniak, "Application of a computer to improve the results obtained in working with the SuperMemo method," 1990.
URL: https://www.supermemo.com/en/archives1990-2015/english/ol/sm2.

[3] "Anki - powerful, intelligent flashcards."
URL: https://apps.ankiweb.net/.

[4] "Semantic UI."
URL: https://semantic-ui.com/.

[5] "Chart.js | Open source HTML5 Charts for your website."
URL: https://www.chartjs.org/.

[6] "Charts | Google Developers."
URL: https://developers.google.com/chart.

[7] "Google charts in tabs - chart size and position incorrect in hidden tab."
URL: https://stackoverflow.com/questions/20290314/.

[8] H.A.Sturges, "The choice of a class interval," *J. American Statistical Association*, pp. 65–66, 1926.
URL: https://bit.ly/2Uy0H6m.

[9] "JQuery - fancytable."
URL: https://www.npmjs.com/package/jquery.fancytable.

Apéndices

A

Datos tablas

A.1. Tablas para superusuarios

En la tabla A.1 se listan los campos que se muestran en las tablas de las diferentes secciones para la vista de administrador, con los significados de dichos campos.

Conjunto	Abreviatura	Descripción
Cursos	name	Nombre del curso
	nuts	Número de unidades
	nqts	Número de preguntas
	ntest	Número de preguntas de tipo test
	nopen	Número de preguntas abiertas
	nans	Número de respuestas
	ntags	Número de etiquetas
	nts	Número de preguntas de tipo test con solución
	pts	Porcentaje de preguntas de tipo test con solución
	mfq	Media de la calidad de las primeras respuestas
	mlq	Media de la calidad de las últimas respuestas
	mfqt	Media de la calidad de la primera respuesta de las preguntas de tipo test
	mlqt	Media de la calidad de la última respuesta de las preguntas de tipo test
	mfqo	Media de la calidad de la primera respuesta de las preguntas de tipo abiertas
Continúa en siguiente página		

Tabla A.1: Información a mostrar para los administradores

Conjunto	Abreviatura	Descripción
	mlqo	Media de la calidad de la última respuesta de las preguntas de tipo abiertas
	mlts	Media de la calidad de la última respuesta de las preguntas de tipo test con solución
	mlnts	Media de la calidad de la última respuesta de las preguntas de tipo test sin solución
	mleas	Media de la facilidad de las últimas respuestas
	mlrep	Media de las repeticiones de las últimas respuestas (parámetro del algoritmo)
	mlint	Media de los intervalos de las últimas respuestas
Etiquetas	tag	Etiqueta
	nans	Número de respuestas
	nq	Número de preguntas
	nopen	Número de preguntas abiertas
	ntest	Número de preguntas de tipo test
	mrep	Media de repeticiones (veces que aparecen las preguntas, no el parámetro del algoritmo)
	eas	Facilidad media de las respuestas
	mfq	Media de la calidad de las primeras respuestas
	mlq	Media de la calidad de las últimas respuestas
	mfqt	Media de la calidad de la primera respuesta de las preguntas de tipo test
	mlqt	Media de la calidad de la última respuesta de las preguntas de tipo test
	mfqo	Media de la calidad de la primera respuesta de las preguntas de tipo abiertas
	mlqo	Media de la calidad de la última respuesta de las preguntas de tipo abiertas

Continúa en siguiente página

Tabla A.1: Información a mostrar para los administradores

Conjunto	Abreviatura	Descripción
	mlint	Media de los intervalos de las últimas respuestas
	mleas	Media de la facilidad de las últimas respuestas
	mlrep	Media de las repeticiones de las últimas respuestas (parámetro del algoritmo)
	noap	Número de respuestas con apuntes entre las de tipo abierto
	nonap	Número de respuestas sin apuntes entre las de tipo abierto
	poap	Porcentaje de respuestas con apuntes entre las de tipo abierto
Estudiantes	user	Usuario
	unit	Unidad actual
	nq	Número de preguntas distintas respondidas
	nans	Número de respuestas
	mrep	Media de repeticiones (veces que aparecen las preguntas, no el parámetro del algoritmo)
	ndays	Número total de días en que se han respondido preguntas
	npd	Número de respuestas por día
	nweeks	Número total de semanas en que se han respondido preguntas
	mfq	Media de la calidad de las primeras respuestas
	mlq	Media de la calidad de las últimas respuestas
	mfqt	Media de la calidad de la primera respuesta de las preguntas de tipo test
	mlqt	Media de la calidad de la última respuesta de las preguntas de tipo test

Continúa en siguiente página

Tabla A.1: Información a mostrar para los administradores

Conjunto	Abreviatura	Descripción
	mfqo	Media de la calidad de la primera respuesta de las preguntas de tipo abiertas
	mlqo	Media de la calidad de la última respuesta de las preguntas de tipo abiertas
	mleas	Media de la facilidad de las últimas respuestas
	mlrep	Media de las repeticiones de las últimas respuestas (parámetro del algoritmo)
	mlint	Media de los intervalos de las últimas respuestas
	noap	Número de respuestas con apuntes entre las de tipo abierto
	nonap	Número de respuestas sin apuntes entre las de tipo abierto
	poap	Porcentaje de respuestas con apuntes entre las de tipo abierto
Preguntas	text	Texto de la pregunta
	id	Identificador de la pregunta
	type	Tipo de pregunta
	sol	¿Tiene solución?
	nans	Número de respuestas
	mrep	Media de repeticiones (veces que aparecen las preguntas, no el parámetro del algoritmo)
	mfq	Media de la calidad de las primeras respuestas
	mlq	Media de la calidad de las últimas respuestas
	mlint	Media de los intervalos de las últimas respuestas
	mleas	Media de la facilidad de las últimas respuestas
	mlrep	Media de las repeticiones de las últimas respuestas (parámetro del algoritmo)

Continúa en siguiente página

Tabla A.1: Información a mostrar para los administradores

Conjunto	Abreviatura	Descripción
	noap	Número de respuestas con apuntes entre las de tipo abierto
	nonap	Número de respuestas sin apuntes entre las de tipo abierto
	poap	Porcentaje de respuestas con apuntes entre las de tipo abierto
Objetivos	id	Id
	unit	Unidad
	obj	Objetivo
	nacc	Número de accesos a los objetivos
	nupl	Número de subidas en los objetivos
	nans	Número de respuestas en los objetivos
	pcorr	Porcentaje de acierto de las respuestas

Tabla A.1: Información a mostrar para los administradores

Algunas aclaraciones sobre la información que se muestra:

- Cuando se habla de "respuestas con apuntes", se refiere, para las preguntas abiertas, a si el usuario ha escrito algo en el campo de texto que hay disponible. Esto permite analizar si el estudiante intenta razonar por escrito su respuesta antes de compararla con la solución, o si prefiere simplemente pensarla.
- El algoritmo de repaso espaciado tiene un parámetro *repeticiones*, que permite establecer un número de veces que la pregunta se debe repetir durante su aprendizaje antes de que se permitan valores grandes del intervalo hasta la siguiente repetición. Este parámetro vuelve a ser cero cuando la pregunta se falla (o se autoevalúa como "Difícil"), por lo que no indica realmente el número de veces que se ha realizado la pregunta. Por esto, en las tablas de información, se distingue entre las repeticiones del algoritmo y las repeticiones reales de la pregunta.
- En la tabla de objetivos, se permite agrupar los datos por unidades, mostrando los valores para el conjunto de objetivos que conforman la unidad. En este caso se omite la columna *Objetivo (obj)*.

A.2. Tablas para estudiantes

En la tabla A.2 se muestran los valores que pueden ver los estudiantes respecto a su información. El administrador también puede acceder a esta información cuando filtra por un usuario concreto.

Aclaración: cuando se habla de "acierto en las respuestas", esto se corresponde de manera trivial con acertar o fallar las preguntas cerradas. En cuanto a las preguntas abiertas, autoevaluarse con

Conjunto	Abreviatura	Descripción
Cursos	name	Nombre del curso
	unit	Unidad
	nans	Número de respuestas
	ndays	Número total de días en que se han respondido preguntas
	npd	Número de respuestas por día
	mfr	Porcentaje de acierto de las primeras respuestas
	mlr	Porcentaje de acierto de las últimas respuestas
Etiquetas	tag	Etiqueta
	nans	Número de respuestas
	mfr	Porcentaje de acierto de las primeras respuestas
	mlr	Porcentaje de acierto de las últimas respuestas
Preguntas	text	Texto de la pregunta
	type	Tipo de pregunta
	nans	Número de respuestas
	fr	Acierto de la primera respuesta
	lr	Acierto de la última respuesta
Objetivos	id	Id
	unit	Unidad
	obj	Objetivo
	nacc	Número de accesos a los objetivos
	nupl	Número de subidas en los objetivos
	nans	Número de respuestas en los objetivos
	pcorr	Porcentaje de acierto de las respuestas

Tabla A.2: Información a mostrar para los estudiantes

“Fácil” se considera acertar, mientras que los otros valores se consideran fallar. De esta forma, acertar se corresponde con obtener el mejor resultado posible, independientemente del tipo de pregunta.

B

Detalles de implementación

B.1. Gestión de la información

A continuación se explica cómo se ha implementado el procesamiento de la información del módulo de estadísticas. Esto pretende servir de guía futura para poder leer el código ya existente y poder realizar cambios de ser necesarios.

Se ha establecido un determinado formato con el que recibir la información de las distintas pestañas. Esto ha permitido crear un template genérico, *table.html*, al que se le llama desde las distintas pestañas con la información particular que se quiere mostrar. Por otro lado, el código para la generación de histogramas también aprovecha este formato.

B.1.1. Guardado de los datos a mostrar

Los datos se guardan en diccionarios Python con varios niveles de profundidad.

La estructuración sigue el prototipo mostrando en el código B.1. En el primer nivel, se usan los identificadores de cursos como clave, después, los identificadores de los objetos, y por último, los pares de clave-valor con los datos concretos. Las claves del último nivel son las que se referencian en la información de las columnas, que se explica en el apartado posterior.

Código B.1: Prototipo diccionario de información

```
1 { course_id : { item_id : { col1 : value1, col2 : value2 } } }
```

Por ejemplo para la pestaña de estudiantes, se generaría un objeto como se muestra en el código B.2. En este ejemplo, para el curso con identificador 9 tenemos datos de tres estudiantes (esto depende de cómo se hayan filtrado los datos), de identificadores 250, 313 y 472. Para cada uno de ellos, se muestra una serie de parámetros que han sido calculados con la información disponible. Por ejemplo, el usuario 313 tiene un valor de 1296 para *nans* (número de respuestas). Esto significa que

ha respondido preguntas de ese curso 1296 veces. Por otro lado, para el curso con identificador 10, hay información nuevamente del usuario 250. Esto significa que está inscrito en varios cursos, y se muestra la información separada de los datos de este usuario para este otro curso.

Código B.2: Ejemplo de diccionario de información

```
{
    9: {
        250: {'nq': 7, 'nans': 7, 'mfq': 2.71, 'mlq': 2.71, 'mlint': 1.0, 'mleas': 2.24, 'mlrep': 0.71, 'nstd': 1,
              'mfqt': 1.0, 'mlqt': 1.0, 'mfqo': 4.0, 'mlqo': 4.0, 'noap': 0, 'nonap': 4, 'poap': 0.0, 'unit': 1,
              'ndays': 1, 'nweeks': 1, 'user': 'eduardo.ricom@estudiante.uam.es', 'npd': 7.0, 'unit_prog':
              '1/92', 'mrep': 1.0},

        313: {'nq': 280, 'nans': 1296, 'mfq': 4.06, 'mlq': 4.58, 'mlint': 30.86, 'mleas': 2.52, 'mlrep': 3.39,
              'nstd': 1, 'mfqt': 3.19, 'mlqt': 4.23, 'mfqo': 4.75, 'mlqo': 4.86, 'noap': 561, 'nonap': 18, 'poap':
              96.89, 'unit': 37, 'ndays': 26, 'nweeks': 8, 'user': 'asier.j@estudiante.uam.es', 'npd': 49.84,
              'unit_prog': '37/92', 'mrep': 4.62},

        472: {'nq': 41, 'nans': 140, 'mfq': 3.21, 'mlq': 4.17, 'mlint': 14.70, 'mleas': 2.17, 'mlrep': 2.48, 'nstd':
              1, 'mfqt': 3.5, 'mlqt': 4.44, 'mfqo': 3.0, 'mlqo': 3.95, 'noap': 17, 'nonap': 63, 'poap': 21.25,
              'unit': 5, 'ndays': 7, 'nweeks': 6, 'user': 'saul@estudiante.uam.es', 'npd': 20.0, 'unit_prog':
              '5/92', 'mrep': 3.41}
    },

    10: {
        250: {'nq': 119, 'nans': 613, 'mfq': 4.72, 'mlq': 4.74, 'mlint': 95.58, 'mleas': 2.72, 'mlrep': 4.51, 'nstd':
              1, 'mfqt': 4.78, 'mlqt': 4.78, 'mfqo': 4.68, 'mlqo': 4.72, 'noap': 55, 'nonap': 307, 'poap': 15.19,
              'unit': 24, 'ndays': 53, 'nweeks': 14, 'user': 'eduardo.ricom@estudiante.uam.es', 'npd': 11.56,
              'unit_prog': '24/24', 'mrep': 5.15}
    }
}
```

La información de todas las pestañas salvo la de cursos se gestiona de la manera que se ha explicado. En el caso de los cursos, se omite el nivel intermedio: para cada curso hay una serie de parámetros que mostrar.

Código B.3: Prototipo diccionario de información para cursos

```
{ course_id : { col1 : value1, col2 : value2 } }
```

B.1.2. Columnas a mostrar

Para decidir qué columnas mostrar en las tablas y los histogramas, se dispone, además del diccionario con los datos, de una lista de diccionarios con los datos de las columnas. Para cada una de ellas se define la abreviatura a mostrar, la descripción detallada (se muestra al pasar el cursor por la cabecera de una tabla), y el tipo de dato. Se muestra un ejemplo en el código B.4.

Código B.4: Ejemplo de lista de columnas

```
[
    {'full_name': 'Usuario', 'abbr': 'user'},
    {'full_name': 'Unidad_actual', 'abbr': 'unit', 'type': 'int'},
    {'full_name': 'Número_de_preguntas_distintas_respondidas', 'abbr': 'nq', 'type': 'int'},
    {'full_name': 'Número_de_respuestas', 'abbr': 'nans', 'type': 'int'},
    {'full_name': 'Media_de_la_calidad_de_las_primeras_respuestas', 'abbr': 'mfq', 'type': 'float'},
    {'full_name': 'Media_de_la_calidad_de_las_últimas_respuestas', 'abbr': 'mlq', 'type': 'float'}
    ...
]
```

La abreviatura (abbr) es la clave por la que buscar en el diccionario de datos, que se ha explicado en el apartado anterior. Esto permite que el código de generación de las tablas sea el mismo en todos los casos, con lo cual se puede cambiar la información del diccionario de datos y de las columnas sin tener cambiar la forma en que se lee. Además, esto facilita que se muestren distintas columnas según el tipo de usuario.

El tipo de dato (type) permite establecer cómo mostrar la información:

- **int**: número entero.
- **float**: número decimal, hace que el número se muestre con exactamente dos decimales.
- **lower**: para cadenas de caracteres, hace que se muestre en minúsculas.
- **boolean**: hace que el valor se muestre con un icono dependiendo de si el valor se puede evaluar como verdadero o falso.

Si un dato se especifica como entero o decimal, su información se mostrará adicionalmente en forma de histogramas.

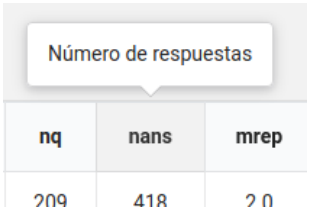
La información detallada (full_name) se muestra al pasar el cursor por encima de los datos de cabecera de las tablas, al pasar el cursor por encima de un histograma, y en la ventana que muestra la información detallada de una barra de histograma. Se muestran ejemplos en la figura B.1.

B.1.3. Realización de consultas

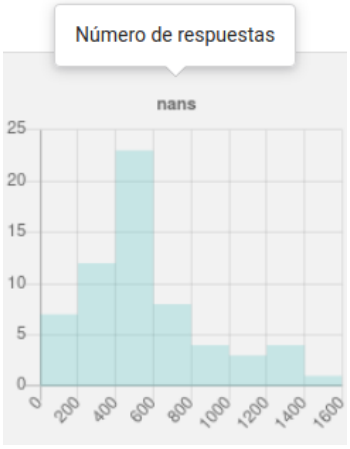
A la hora de realizar consultas, casi siempre se va a seguir un patrón: filtrar la información, agrupar dependiendo del contexto, y anotar valores de interés.

Se ha hecho que, en la medida de lo posible, los filtros, agrupadores y valores a anotar se declaren de forma separada de las consultas, para que dichas consultas tengan un código más sencillo. Se muestra un ejemplo simplificado en el código B.5.

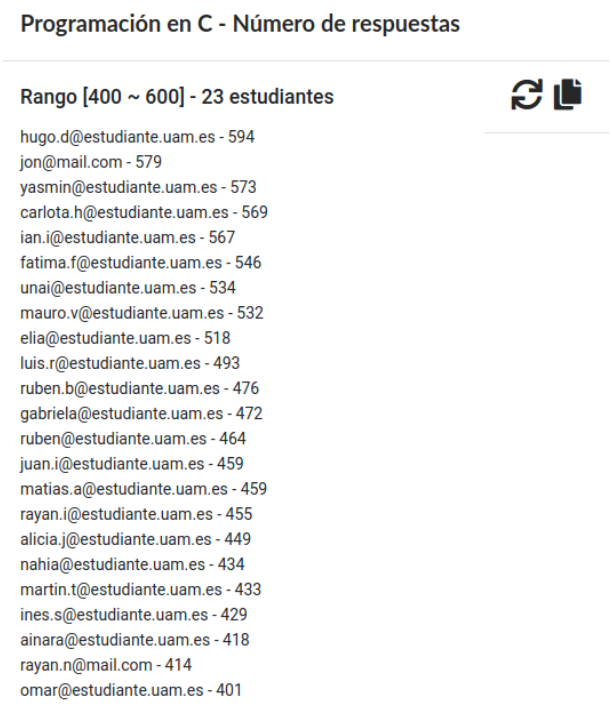
En el ejemplo se muestra la manera en la que se realizan las consultas para la pestaña *Usuarios*. Primero se declaran los parámetros de las consultas: filtrar por usuarios y etiquetas, agrupar por curso



(a) Cabecera de tablas



(b) Histogramas



(c) Detalles de histogramas

Figura B.1: Usos de la descripción de columnas

Código B.5: Ejemplo de consultas generalizadas

```
filters = {
    'user__in' : filter_data['users'],
    'test__course_and_tag__in' : filter_data['courses_and_tags']
}

groupers = ['user', 'test__unit__course']

annotations = {
    'course_id' : F('test__unit__course__id'),
    'user_id' : F('user__id')
}

nans = FlashCardHistory.objects
            .filter(**filters)
            .order_by(*groupers).values(*groupers)
            .annotate(**annotations, nans=Count('pk'))

mq = CacheUserTest.objects.filter(**filters)
            .order_by(*groupers).values(*groupers) \
            .annotate(**annotations, mq=Avg('quality')
```

y usuario, y anotar dichos valores.

Después, se realizan las consultas, y se puede ver que se anotan unos valores adicionales: lo que se espera conseguir de cada consulta particular, que puede ser contar el número de valores o calcular una media. Una vez realizada una consulta, se puede iterar sobre los resultados y transferir los valores a un diccionario Python. Al haber anotado el valor del curso y el usuario, se puede guardar la información en la jerarquía que se ha explicado en el apartado B.1.1.

Dado que varias de estas consultas son iguales para varias pestañas, como *la facilidad de las últimas respuestas*, se ha realizado una función que realiza varias de estas consultas comunes de forma generalizada. Simplemente hay que pasarle un conjunto de parámetros con los que realizar las consultas que variará dependiendo de cómo se quiera mostrar la información en la pestaña correspondiente.

Algunas otras consultas requieren unos parámetros muy concretos y sólo se utilizan para una de las pestañas, por lo que no siempre es necesario realizar consultas de esta manera.

B.2. Actualización de la versión de Django

Para la creación de las tablas, se puede usar simplemente el sistema de templates que ofrece Django, de modo que todos los datos se escriben directamente en el código HTML. Por otro lado, a la hora de realizar gráficas, no siempre se puede hacer de la misma manera. Para un diagrama de barras simple se puede generar el código JavaScript necesario como parte del template, pero en los casos

en los que hay que realizar un tratamiento de los datos más complejo, resulta más complicado. Por ejemplo, para la realización de histogramas, hay que hacer ciertas operaciones para decidir el ancho y alto de las barras.

La solución está en poder crear un objeto de JavaScript a partir de los datos del diccionario Python. Así, cuando la página se haya cargado, se puede iterar sobre los datos y realizar operaciones para decidir cómo mostrar la información en un gráfico.

Django cuenta con una opción *json_script* que permite realizar esta conversión en el template. Esto permite realizar la traducción y evitar problemas con caracteres especiales.

Sin embargo, esta opción está disponible a partir de la versión 2.1, mientras que el proyecto estaba en versión 2.0. Dado que no se trataba de una versión muy “lejana”, se probó a realizar el cambio para contar con esta funcionalidad, y se comprobó que seguía siendo compatible con el resto de la aplicación.

B.3. Gestión de valores atípicos

La gestión de valores atípicos se decidió realizar porque para el curso que se estaba desarrollando en el momento había un estudiante que había utilizado su cuenta varios meses atrás, y hacía que en el gráfico de respuestas del conjunto de la clase aparecieran unas respuestas aisladas de fechas lejanas.

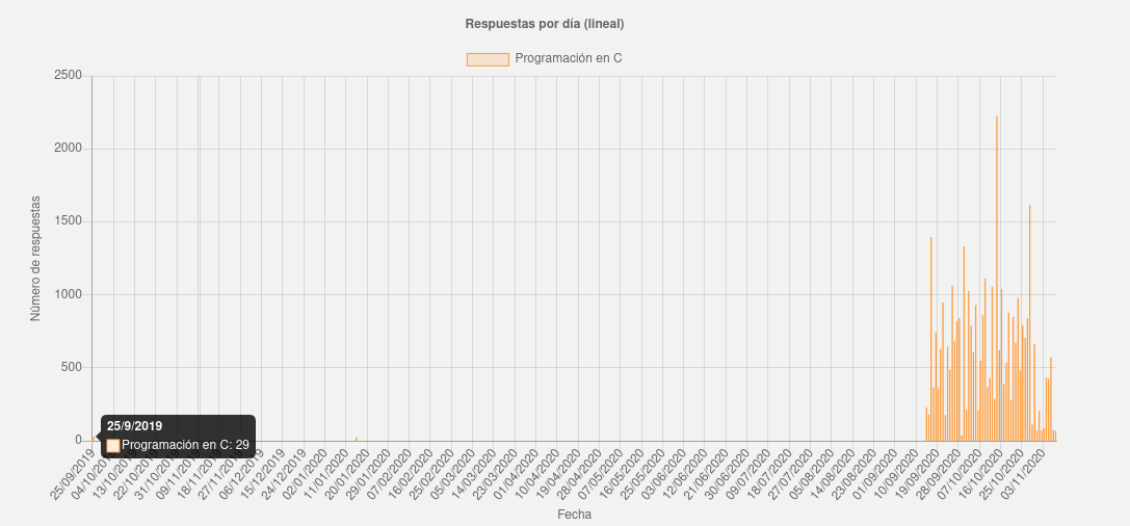


Figura B.2: Ejemplo de valores atípicos

Dado que el módulo de estadísticas no estaba orientado al filtrado por fecha, se decidió buscar una manera de detectar estos valores atípicos y eliminarlos de la gráfica en el momento de ejecución.

Inicialmente se probó a realizar una implementación basada en percentiles, que permite detectar los valores lejanos a la mayoría. Esto se hace normalmente para valores atípicos en el eje de abscisas,

pero para este caso, se buscan valores atípicos en el eje de coordenadas. Esto se hace así porque si se dispone de una serie de valores seguidos, no conviene eliminar los valores más bajos, sobre todo si está en una zona intermedia del conjunto, lo cual puede pasar. En su lugar, se buscan las fechas aisladas de la mayoría del conjunto.

Para el ejemplo que se ha mostrado en la figura B.2, funciona correctamente, dado que tiene un par de valores aislados del conjunto, y se quitan sin problema, como se ve en la figura B.3.

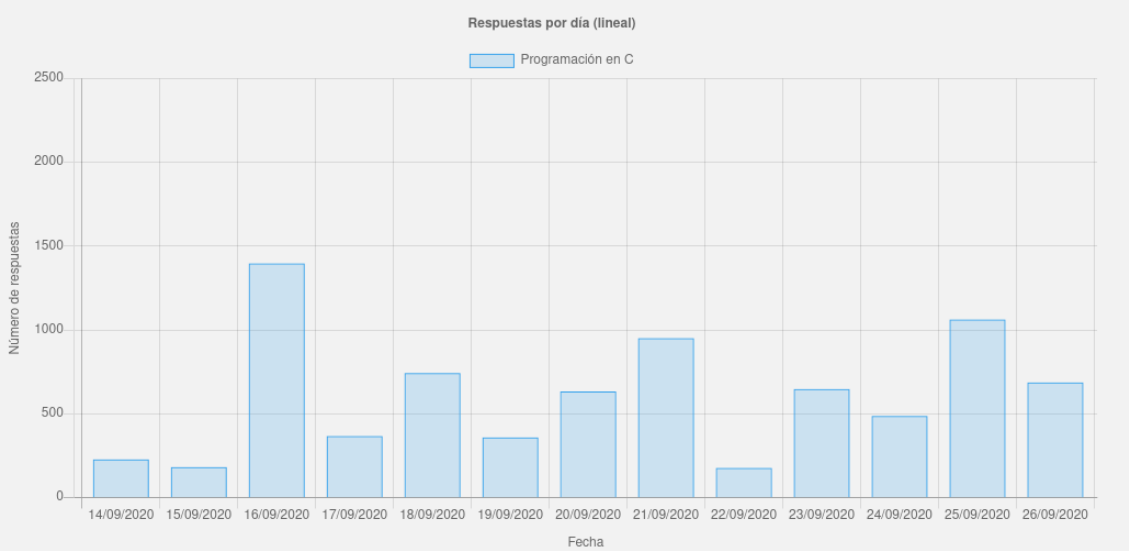


Figura B.3: Intento de exclusión de valores atípicos

Sin embargo, este método no funciona bien con otros conjuntos de datos. Por ejemplo, en la figura B.4, con datos reales hasta septiembre de 2020, se muestran todos los datos disponibles de *Programación en C*, sin aplicar filtros de estudiantes.

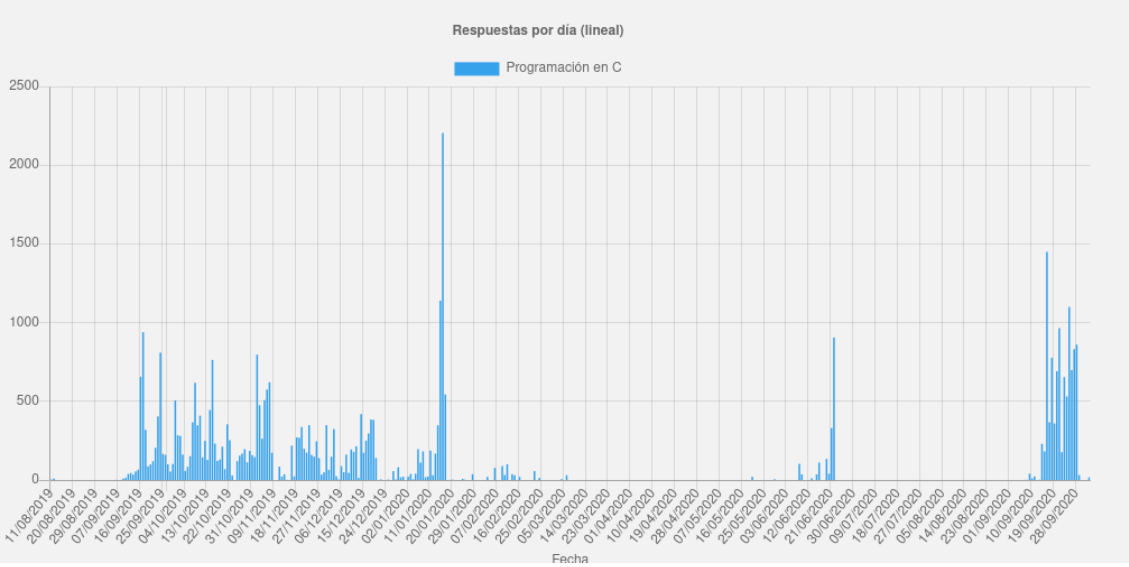


Figura B.4: Otro ejemplo de valores atípicos

Se puede observar que hay una serie de datos de bastante peso de septiembre de 2019 hasta final del año, primer curso en el que se utiliza la aplicación. Después se puede ver que en septiembre de 2020 el número de respuestas vuelve a aumentar considerablemente dado el comienzo de un nuevo curso, aunque a falta de más tiempo para que tenga tanto peso como el curso anterior.

Al aplicar la fórmula de percentiles, como se muestra en la figura B.5, se observa que se muestran principalmente los datos del curso anterior, por lo que se dejan de mostrar los datos de septiembre de 2020, que son relevantes. Además, se puede observar que en la parte izquierda de la gráfica hay un número pequeño de respuestas y un hueco de varios meses.

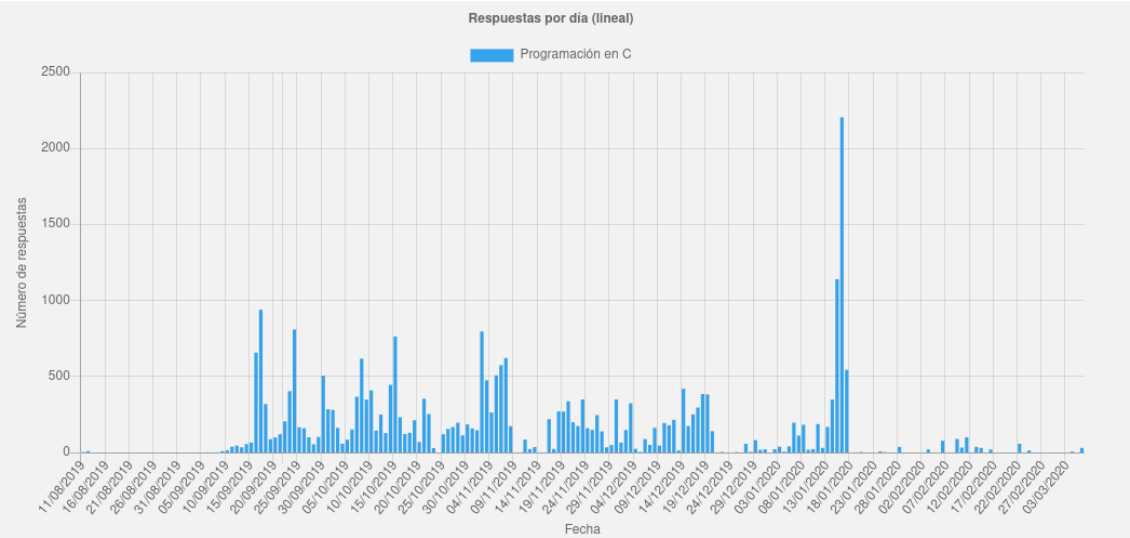


Figura B.5: Intento fallido de exclusión de valores atípicos

Dado que la fórmula de percentiles no siempre muestra datos relevantes, se ha decidido implementar una solución más manual. Lo que se hace es recorrer los datos tanto desde el principio como desde el final. Al hacerlo, se buscan series de datos seguidos que tengan al menos un elemento con un valor mayor igual a la media. Una vez que se encuentra, se tiene en cuenta el primer valor encontrado de la serie.

En la figura B.6 se muestra un ejemplo gráfico. Se ha marcado con flechas los puntos iniciales de las series de valores (a veces valores sueltos), y se muestran en verde los que se consideran válidos, al estar en una serie con algún valor mayor de la media. Estos valores conforman el mínimo y el máximo del resultado final.

De esta forma, los casos de fechas aisladas no suelen cumplir la condición, porque suelen tener unas pocas respuestas de pocos estudiantes. Y así se consigue que si hay una serie de fechas seguidas con un número destacable de respuestas, se pueda tomar en consideración, aunque haya sido por un tiempo relativamente corto.

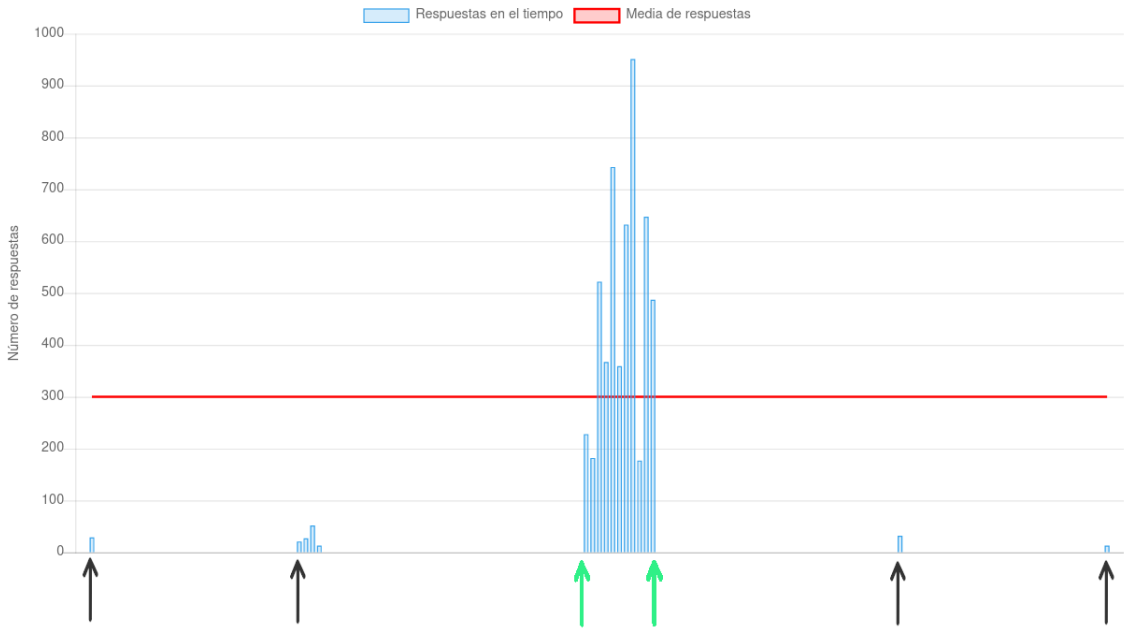


Figura B.6: Método manual de exclusión de valores atípicos

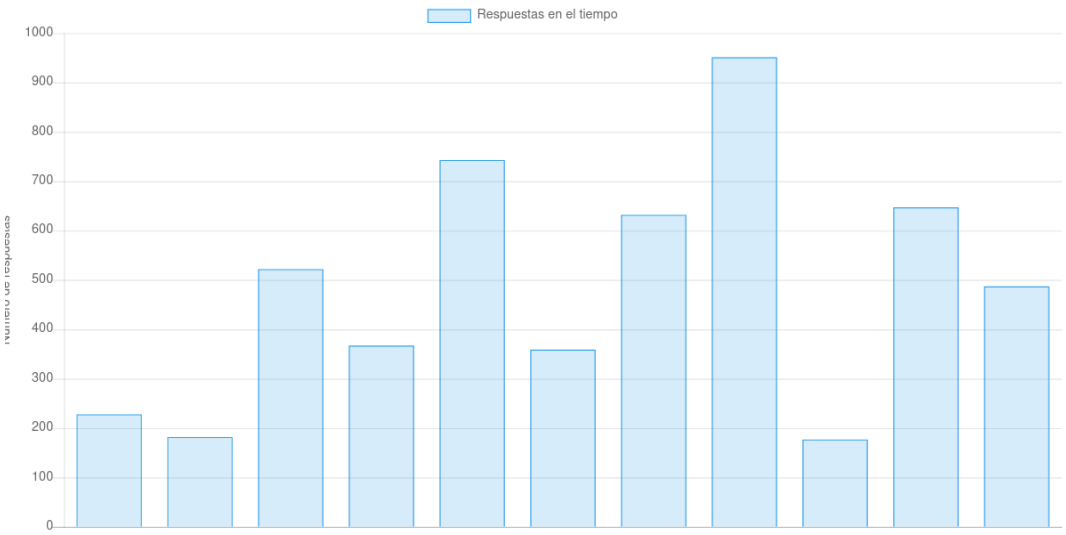


Figura B.7: Resultado de exclusión manual de valores atípicos

B.4. Indicaciones para probar cambios en el futuro

En lo que se refiere a estilo del código en el módulo de estadísticas, se recomienda que se sigan las indicaciones explicadas en el apartado anterior, para simplificar el desarrollo y mantener un código homogéneo en la medida de lo posible.

A la hora de realizar cambios en el código, o incluso si se cambiara cualquier parámetro en la jerarquía de clases, se debe probar que se mantiene la funcionalidad ya existente. A continuación se listan los aspectos que se deberían comprobar:

- Probar el módulo con y sin filtros, y ver que funciona correctamente.
- Probar la funcionalidad de foco de estudiantes y de etiquetas, y ver que se destacan los datos correctos.
- Acceder mediante un usuario normal, que tenga algunas preguntas respondidas, y comprobar que se muestran datos distintos.
- Probar utilizando la vista de estudiante, como se explica en el apartado 4.5.
- Probar la funcionalidad de grupos públicos.

    1.– Crear un grupo que incluya un usuario con el que poder probar, y declarar el grupo público. Si se está probando en el servidor de producción, el grupo debería incluir sólo usuarios de prueba, para evitar que afecte a los usuarios habituales de la aplicación. También se podría añadir un usuario de prueba de forma temporal a un grupo que ya se esté utilizando como público, si hay más de 20 usuarios, dado que no afectará notablemente a los datos medios. En caso de estar en un entorno de pruebas, se puede usar cualquier conjunto de usuarios.

    2.– Probar a filtrar por el usuario concreto, y utilizando la vista de estudiante.

    3.– Por otro lado, acceder como el usuario de prueba, y acceder al módulo.

- Probar funcionalidades adicionales: cambiar la paginación, seleccionar estudiantes de filtros mediante archivo, opciones de edición de grupos.
- Uso de la funcionalidad de descarga para los datos.
    - Si no se han realizado cambios en la manera en que se descargan los datos, basta con comprobar que la descarga se efectúa correctamente. En caso de que haya un error, se puede deber a que se ha realizado un cambio en la forma en que se estructura la información o simplemente que se haya cambiado la clave con la que se guardan ciertos datos en un diccionario. De ser así, debería ajustarse la función de descarga para que se adapte a los datos cambiados.
    - Para las opciones relacionadas con los datos de las tablas que se pueden observar en la aplicación, se recomienda probar usando unos filtros cualesquiera, y comprobar que la información que se descarga se corresponde con la que se visualiza en la aplicación.
    - Para los datos de preguntas por día, se puede comprobar, en el gráfico de la pestaña de *Cursos*, que los valores son equivalentes. Hay que tener en cuenta que la información descargada incluye los valores atípicos, por lo que puede ser necesario utilizar la opción que los incluye en el gráfico.
    - Respecto a la opción de *Incluir FlashCardHistories*, hay que tener en cuenta de que se trata de mucha información, que requiere mucho tiempo de descarga, y no se puede comprobar a menos que se compare directamente con la base de datos. Se recomienda filtrar para un solo usuario, y ver que sigue funcionando correctamente.

Por lo general se recomienda realizar estas pruebas para comprobar que la aplicación no sufre

ningún error en distintos escenarios. Dependiendo de los cambios que se hayan realizado, puede ser necesario comprobar que los datos tengan sentido o que se mantenga cierta coherencia en la información. Por ejemplo, si el administrador accede a la vista de estudiante, debería ver los mismos datos que al estudiante se le muestra al utilizar la aplicación.

C

Desactivación de preguntas

Con vistas al uso a largo plazo de la aplicación, se ha añadido en los objetos de la clase StudentCourse un campo *show_questions* de tipo booleano (por defecto verdadero) que decide si mostrar al usuario preguntas diarias de ese curso. Esto se hace porque los estudiantes que hayan hecho *Programación en C* no querrán seguir recibiendo las mismas preguntas en el futuro si hacen *Programación en Kotlin* usando la misma cuenta. Eso sí, el estudiante no puede desactivar las preguntas, sino que tiene que ponerse en contacto con el administrador para que se lo desactive.

Se ha personalizado la interfaz de administración para la clase StudentCourse para que se pueda escoger estudiantes y ejecutar una acción que cambie el valor de *show_questions*.

De cara a no mostrar preguntas, en el controlador del módulo tutorial (fichero views.py), cada vez que se filtran los objetos FlashCard para ver los tests pendientes, se filtra además por los StudentCourse del estudiante que tenga el booleano *show_questions* como verdadero. Los únicos casos en los que no se ha hecho ha sido cuando se filtra por una unidad específica o un test específico.

Este cambio no era necesario para el módulo realizado, pero se ha aprovechado a realizar ahora para que pueda ser útil en el futuro.

D

Jerarquía de clases

Con la realización del nuevo módulo, se han creado varias clases para apoyar la funcionalidad y se han realizado unos cambios menores en las clases preexistentes. En esta sección se pretende sintetizar el estado actual de las clases para que pueda servir de referencia para el trabajo futuro de la aplicación.

En la figura D.1 se muestra un esquema general de las clases y la forma en que se relacionan.

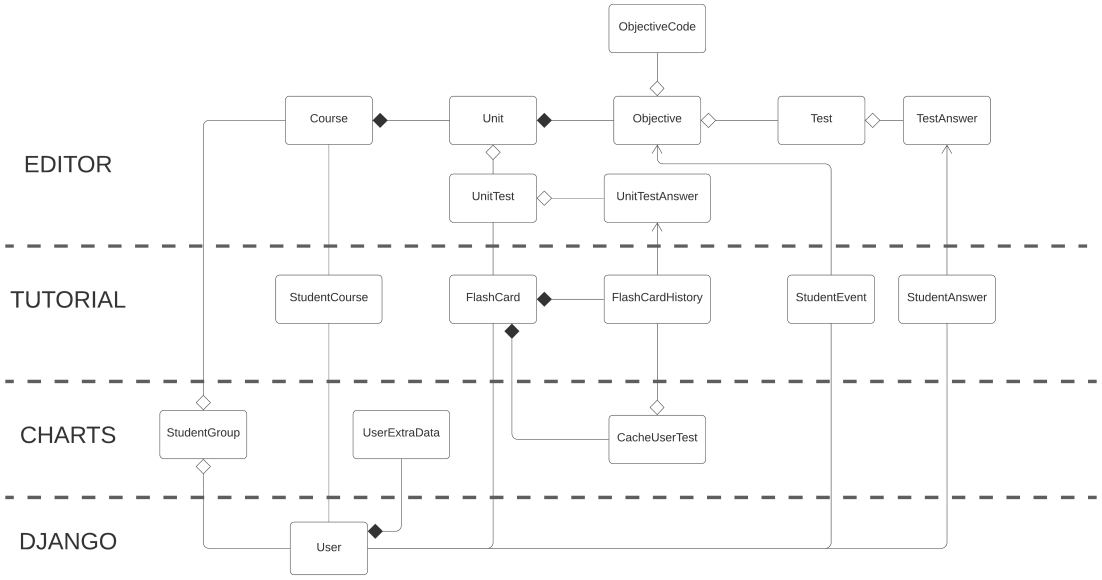


Figura D.1: Relaciones entre clases al terminar el proyecto

En la figura D.2 se muestran las clases con sus campos y las referencias a otras clases. Se destacan en otro color las clases nuevas y se marcan en negrita los campos nuevos en clases preexistentes. No se muestra la clase User, dado que se utiliza muchas veces, para simplificar el diagrama. Todos los campos *user* corresponden con una referencia a tal clase, y el campo *students* de la clase StudentGroup consiste en una o varias referencias a User.

Por último, en las tablas D.1, D.2 y D.3 se incluyen las clases de los distintos módulos junto con una descripción de todos sus campos.

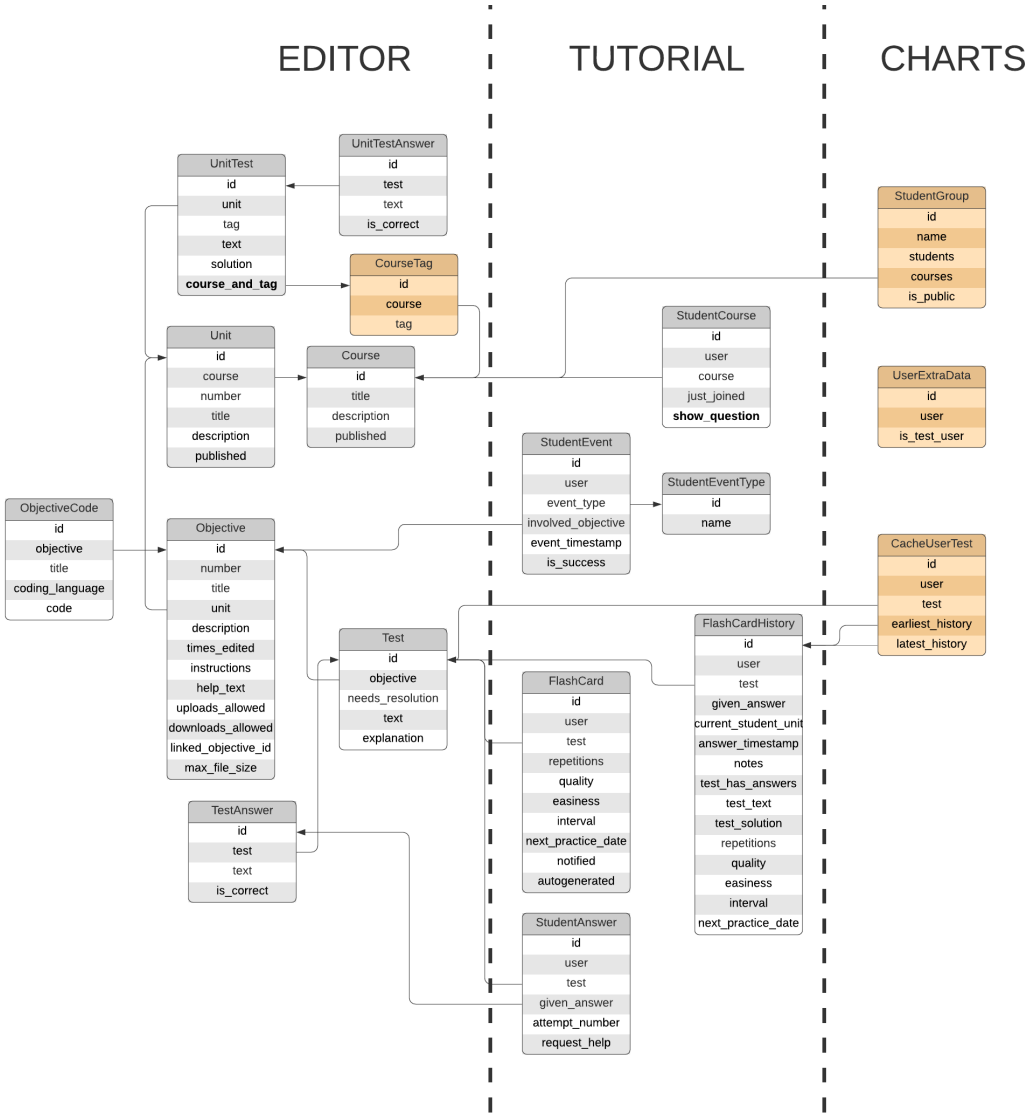


Figura D.2: Jerarquía de clases con campos y referencias al terminar el proyecto

Clase	Atributos
Course	• **title**: Nombre del curso. • **description**: Breve descripción del curso. • **published**: Determina si los usuarios pueden acceder a los contenidos del curso.
Unit	• **course**: Curso al que la unidad pertenece. • **number**: Número de unidad dentro del curso, empezando por 1. • **title**: Nombre de la unidad. • **description**: Breve descripción de la unidad. • **published**: Determina si los usuarios pueden acceder a los contenidos de la unidad.
Objective	• **unit**: Unidad a la que el objetivo pertenece. • **number**: Número de objetivo en la unidad, empezando por 1. • **title**: Nombre del objetivo. • **description**: Descripción del objetivo. • **times_edited**: Número de veces que un administrador ha editado el contenido del objetivo, después de ser creado. • **instructions**: Enunciado principal del objetivo, en formato HTML. • **help_text**: Texto de ayuda para ayuda a comprender el objetivo, en formato HTML. • **uploads_allowed**: Determina si el objetivo incluye una subida de archivos. En caso de haberla, su realización es obligatoria. • **downloads_allowed**: Determina si se pueden descargar ficheros. • **linked_objective_id**: Identificador del objetivo con los ficheros a descargar. • **max_file_size**: Tamaño máximo de las subidas de archivos en MB.
Continúa en siguiente página	

Tabla D.1: Clases del módulo Editor

Clase	Atributos
ObjectiveCode	• **objective**: Objetivo en el que incluir el código. • **number**: Número de pestaña en la que mostrar el código, empezando por 1. • **title**: Título de la pestaña que mostrar en el tutorial. • **coding_language**: Lenguaje de programación en el que está escrito el código. • **code**: Código a mostrar.
Test	• **objective**: Objetivo en el que incluir la pregunta. • **needs_resolution**: Determina si se requiere una respuesta. Actualmente todos los *Test* requieren respuesta. • **text**: Contenido de la pregunta, en formato HTML. • **resolution**: Explicación de la solución o ejemplo de solución propuesta, en formato HTML.
TestAnswer	• **test**: Ejercicio (*Test*) al que pertenece la posible respuesta. • **text**: Texto de la respuesta. • **is_correct**: Determina si es la respuesta correcta a la pregunta.
CourseTag	• **course**: Curso relacionado. • **tag**: Texto de la etiqueta relacionada.
Continúa en siguiente página	

Tabla D.1: Clases del módulo Editor

Clase	Atributos
UnitTest	• **unit**: Unidad a la que pertenece la pregunta. • **tag**: Texto de la etiqueta de la pregunta. • **text**: Contenido de la pregunta, en formato HTML. • **solution**: Explicación de la solución o ejemplo de solución propuesta, en formato HTML. • **course_and_tag**: Par de curso y etiqueta al que pertenece la pregunta.
UnitTestAnswer	• **test**: Ejercicio (*UnitTest*) al que pertenece la posible respuesta. • **text**: Texto de la respuesta. • **is_correct**: Determina si es la respuesta correcta a la pregunta.

Tabla D.1: Clases del módulo Editor

Clase	Atributos
StudentCourse	• **user**: Usuario inscrito. • **course**: Curso al que el usuario está inscrito. • **just_joined**: Determina si el usuario se acaba de inscribir y todavía no ha accedido al curso. • **show_questions**: Determina si el usuario recibe notificaciones de preguntas tipo UnitTest para el curso.
StudentAnswer	• **user**: Usuario que ha realizado la respuesta. • **test**: Pregunta respondida (*Test*). • **given_answer**: Respuesta escogida (*TestAnswer*). • **attempt_number**: Número de intento en el que se responde la pregunta. • **requested_help**: Determina si el usuario consultó la ayuda disponible para la resolución del ejercicio.
Continúa en siguiente página	

Tabla D.2: Clases del módulo Tutorial

Clase	Atributos
FlashCard	• **user**: Usuario que ha realizado la respuesta • **test**: Pregunta relacionada (*UnitTest*). • **repetitions**: Número de repeticiones de la pregunta durante su etapa de aprendizaje. • **quality**: Valor numérico que representa el resultado de la última respuesta, tanto si es abierta como tipo test. • **easiness**: Valor que representa la facilidad de la pregunta para el usuario. Este valor se actualiza a cada respuesta y sirve de multiplicador para calcular el nuevo valor del intervalo. • **interval**: Número de días desde la última realización de la pregunta hasta su siguiente repetición. • **next_practice_date**: Fecha a partir de la cual se podrá repetir la pregunta. • **notified**: Determina si el usuario ha sido notificado de que puede realizar la pregunta de esta tarjeta (porque haya llegado la fecha de realización) pero ha decidido posponer su realización. • **autogenerated**: Determina si la tarjeta se ha generado automáticamente, en caso de que el usuario complete una unidad pero no llegue a responder una pregunta relacionada.
Continúa en siguiente página	

Tabla D.2: Clases del módulo Tutorial

Clase	Atributos
FlashCardHistory	• **user**: Usuario que ha realizado la respuesta • **test**: Pregunta relacionada (*UnitTest*). • **given_answer**: Respuesta escogida (*UnitTestAnswer*) en caso de que fuera una pregunta tipo test. • **current_student_unit**: Unidad máxima alcanzada por el estudiante en el curso relacionado con la pregunta, en el momento de su realización. • **answer_timestamp**: Marca de tiempo de la realización de la respuesta. • **notes**: Texto escrito por el usuario en el campo de texto disponible en las preguntas abiertas. • **test_has_answers**: Determina si la pregunta tiene respuestas posibles, es decir, si es una pregunta tipo test o no. • **test_text**: Contenido de la pregunta, en formato texto. • **test_solution**: Explicación de la solución o ejemplo de solución propuesta, en formato texto. • **repetitions**: Valor del campo *repetitions* de la tarjeta tras realizar la respuesta. • **quality**: Valor del campo *quality* de la tarjeta tras realizar la respuesta. • **easiness**: Valor del campo *easiness* de la tarjeta tras realizar la respuesta. • **interval**: Valor del campo *interval* de la tarjeta tras realizar la respuesta. • **next_practice_date**: Valor del campo *next_practice_date* de la tarjeta tras realizar la respuesta.
Continúa en siguiente página	

Tabla D.2: Clases del módulo Tutorial

Clase	Atributos
StudentEvent	• **user**: Usuario que ha generado el evento. • **event_type**: Tipo de evento. Puede ser 'OBJECTIVE_ACCESS', 'TEST_ANSWER' o 'FILE_UPLOAD'. • **involved_objective**: Objetivo sobre el cual se ha generado el evento. • **event_timestamp**: Marca de tiempo del evento. • **is_success**: Si el evento ha tenido lugar con éxito. Sólo se utiliza para eventos tipo 'TEST_ANSWER', para determinar si se ha dado una respuesta correcta o no.

Tabla D.2: Clases del módulo Tutorial

Clase	Atributos
StudentGroup	• **name**: Nombre del grupo. • **students**: Usuarios pertenecientes al grupo, debiendo haber al menos uno. • **courses**: Cursos relacionados con el grupo. Puede no tener ninguno. • **is_public**: Determina si los usuarios pertenecientes pueden ver algunos datos medios en las estadísticas, para los cursos del grupo.
UserExtraData	• **user**: Usuario relacionado. • **is_test_user**: Si el usuario es de prueba, permitiendo excluirlo en el módulo de estadísticas.
CacheUserTest	• **user**: Usuario que ha realizado la respuesta • **test**: Pregunta relacionada (*UnitTest*). • **earliest_history**: Registro de la primera respuesta realizada (*FlashCardHistory*). • **latest_history**: Registro de la última respuesta realizada (*FlashCardHistory*).

Tabla D.3: Clases del módulo Charts


UAM
UNIVERSIDAD AUTONOMA
DE MADRID